Instituto Tecnológico
y de Estudios Superiores de Occidente
Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Maestría en Sistemas Computacionales

Evaluación y Punto de Referencia para Bases de Datos Orientadas a Grafos

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN SISTEMAS COMPUTACIONALES

Presenta: **LEOBARDO RUIZ ROUNTREE**

Asesor **MTRO. VÍCTOR HUGO ORTEGA GUZMÁN**

Tlaquepaque, Jalisco. Junio de 2019.

AGRADECIMIENTOS

Quiero agradecer a mi familia, en especial a mi esposa Valeria e hijo Emiliano, quienes, a mi lado, han cursado esta etapa importante de mi desarrollo personal y profesional, ustedes han aportado un elemento clave en este trabajo, la inspiración. Gracias, los amo. Mamá, Papá, Gol, y Daniel sé que en reuniones me veían agotado, lo estaba, pero también me ayudaron a despejar mi mente en momentos de agobio, muchas gracias, los amo. A Tomasa y Chunior, sé que no pueden leer esto y no entienden que está pasando, así que solo les diré esto: guau guau guau guau, gracias por sentarse a mi lado toda noche.

Deseo dar las gracias al Maestro Víctor Hugo Ortega Guzmán por el asesoramiento brindado durante el desarrollo de este trabajo; a los docentes que impartieron las materias a lo largo de la Maestría en Sistemas Computacionales; al CONACYT por la beca otorgada, con el apoyo número 487248; al ITESO por el descuento aplicado a través de los convenios empresariales y proveer de la infraestructura adecuada para cursar la maestría; a Intel por apoyarme con el tiempo que requerí y apoyos económicos durante estos 2 años.

DEDICATORIA

“El precio del éxito es trabajo duro, dedicación y determinación de que, ganes o pierdas, habrás hecho todo lo que estaba en tus manos” (Vince Lombadi).

De mis padres aprendí que nada se regala en la vida, todo logro requiere de un esfuerzo individual con soporte de tu entorno. Y es así como trato de seguir el ejemplo que me dieron. El desarrollo de este trabajo es uno de muchos ejemplos en mi vida donde lo he aplicado.

Para Valeria y Emiliano, con ustedes he aprendido que los logros se disfrutan al máximo cuando se adquieren a través de lo colectivo. Cualquier éxito ya sea de pequeña proporción o de grande alcance tiene un sabor diferente en compañía de ustedes, un sabor que se parece al de una refrescante cerveza en una tarde de primavera, o al dulce té negro con limón, ese que tanto te gusta en cualquier circunstancia hijo. Y créanme, este es uno de esos grandes logros.

Para ustedes, por ustedes. Me alentaron y me levantaron siempre que lo necesité…

RESUMEN

Las bases de datos orientadas a grafos (BDG) han adquirido popularidad dentro del análisis de datos masivos ya que proveen un rendimiento superior a la que se obtiene a través de una base de datos relacional en los escenarios en donde la alta conectividad entre datos se convierte en el principal componente para interpretar datos y obtener información de ellos para la toma de decisiones. Este trabajo de obtención de grado (TOG) tiene como objetivo desarrollar una metodología para comparar y evaluar de forma equitativa a distintos motores que se especializan en el manejo de grafos.

En primera instancia se analizan los estudios relacionados a este tema que fungirán como soporte para nuestra investigación e identificar los puntos a mejorar para crear un proceso de evaluación y realizar el punto de referencia de bases de datos orientadas a grafos que son populares por el tiempo que llevan desarrollándose y siendo utilizadas en la industria y la academia, y otros que han surgido recientemente para mejorar las limitantes que hay en el mercado.

El principal componente del trabajo es crear los pasos requeridos para ejecutar pruebas con distintos tipos de conjuntos de datos y algoritmos a un grupo de BDG. De forma posterior se ejecuta un caso de estudio en el cual se define un ambiente de validación homogéneo para todo sistema, donde se puedan tener las especificaciones de hardware y software para que todas las BDG puedan correr sin restricciones. De la misma manera se definen los aspectos a evaluar, que incluyen, pero no están limitados a las capacidades del lenguaje de consultas que proveen, la integración con otras plataformas o sistemas, y el soporte que cada una provee para la ejecución de algoritmos sobre grafos. La selección de los datos que se cargarán en las distintas plataformas debe considerar que tengan el formato adecuado que cada BDG soporta, y en caso contrario, analizar si los datos pueden ser convertidos para ser utilizados en la base de datos.

Finalmente, se toman como caso de prueba las bases de datos basadas en grafos *GraphDB*, *JanusGraph*, *Neo4j*, y *TigerGraph*. El caso de estudio utiliza la metodología desarrollada a través de este trabajo para evaluar las bases de datos.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS Y ABREVIATURAS

ACID		*Atomicity, Consistency, Isolation, Durability*
API		*Application Programming Interface*
AWS		Amazon Web Services
BDG		Base de Datos orientada a Grafos
BFS		*Breath First Search*
BIOS		*Basic Input/Output System*
BMM		*Bipartite Maximal Matching*
BSP		*Bulk Synchronous Parallel*
CPU		*Central Processing Unit*
CSV		*Comma Separated Version*
DAS-4		*Distributed ASCI Supercomputer 4*
DFS		*Depth First Search*
GAS		*Gather Apply Scatter*
GC		*Graph Coloring*
GPS		*Global Positioning System*
IDI		Área de Investigación, Desarrollo e Innovación
HPC		*High Performance Computing*
ITESO		Instituto Tecnológico de Estudios Superiores de Occidente
JSON		*JavaScript Object Notation*
LALP		*Large Adjacency-List Partitioning*
LPG		*Labeled Property Graph*
N/A		*Not Available*
NoSQL		*None SQL (Structured Query Language)*
PR		*PageRank*
RAM		*Random Access Memory*
RDF		*Resource Description Framework*
SATA		*Serial ATA*
SQL		*Structured Query Language*
SNAP		*Stanford Network Analysis Platform*
SSD		*Solid State Disk*
SSSP		*Single Source Shortest Path*
SV		*Shiloac-Viskin*
TOG		Trabajo de Obtención de Grado
URI		*Uniform Resource Identifier*
W3C		*Word Wide Web Consortium*
WCC		*Weakly Connected Components*

1 INTRODUCCIÓN

En la actualidad existen estudios que realizan un análisis comparativo de bases de datos orientada a grafos (BDG), aunque sus resultados son útiles, la mayoría no son comparables, ya sea por utilizar distintos conjuntos de datos, realizar pruebas en ambientes que no son homogéneos entre ejecuciones, tener preguntas de análisis diferentes, o simplemente por tener características que no hacen posible comparar los resultados obtenidos [1].

El objetivo principa; del proyecto es proponer una metodología para realizar un análisis formal de cualquier conjunto de BDG en el mismo entorno de ejecución, con los mismos datos de entrada y respondiendo las mismas preguntas para ofrecer información acerca de las fortalezas, áreas de oportunidad y el escenario idóneo el que se puede utilizar cada una de las herramientas evaluadas.

1.1 Antecedentes

La teoría de grafos es una base importante para resolver diversos problemas tales como el desarrollo de redes (computacionales, sociales, telecomunicaciones, etc.), representación de estructuras de datos, mapeo y descubrimiento de caminos óptimos, modelado de moléculas, inteligencia artificial (satisfacción de restricciones) por mencionar algunas.

Las redes sociales más populares como *Facebook*, *Twitter* e *Instagram* crean las relaciones entre usuarios a través de grafos. Los mapas de geo ubicación como *Waze* y *Google Maps* hacen cálculos de rutas y tiempos estimados de arribo a través de grafos. Ya que cada una de estos ejemplos resuelven problemas a través de BDG, es importante notar que no todas utilizan el mismo motor de base de datos, dado que cada una de las soluciones de bases de datos orientadas a grafos existentes tienen fortalezas y debilidades. Saber desde un principio cuál es el motor ideal para el problema a solucionar es fundamental para el éxito del producto o investigación. La creación de una metodología para el análisis de las BDG proporcionaría datos relevantes para apoyar la toma de decisión sobre el mejor producto para una serie de requerimientos previamente conocidos.

1.2 Justificación

Tanto la academia como la industria tienen la necesidad de utilizar la teoría de grafos para solucionar problemas específicos a través del uso de las propiedades de los mismos. Se necesita saber cuáles de las BDG disponibles en el mercado son las ideales para representar billones de nodos con propiedades entre ellos, mientras que otros están más interesados en la velocidad de procesamiento en búsquedas de datos.

Las razones principales que son mencionadas de forma recurrente entre desarrolladores de software e investigadores que utilizan las ventajas que proveen las BDG, se incluyen estos tópicos:

- Alto desempeño en consultas sobre conjuntos de datos a escala.
- Capacidad de almacenar datos en el orden de petabytes (10^{15}).
- Lenguajes de consultas intuitivos dada su forma natural para representar la conexión entre datos.
- La facilidad con la que se puede adaptar aplicaciones a través del tiempo, esto habilita el desarrollo de software basado en la metodología *agile*.
- Permiten tipos de datos nuevos, se adaptan a datos no estructurados los cuales son muy comunes en datos generados en tiempo real.
- Están optimizadas para operaciones orientadas a minería de datos.

En general, estas características se pueden clasificar en tres áreas: Desempeño, Flexibilidad, y Agilidad.

El desempeño es relevante cuando se manejan las relaciones que existen entre los datos. Las BDG mejoran el desempeño que tienen otro tipo de bases de datos cuando se analizan redes o se hacen consultas que requieren el recorrido total de un grafo. También mejoran el desempeño que tienen las bases de datos relacionales que utilizan el lenguaje SQL (*Structured Query Language*) cuando realizan operaciones relacionales de tipo join para entender cómo se relacionan los datos entre ellos en redes muy grandes.

La flexibilidad de los esquemas de las BDG permite que los arquitectos y diseñadores de sistemas puedan diseñar esquemas que se adaptan de manera fácil a cambios importantes cuando las necesidades del negocio comienzan a evolucionar. En lugar de modelar dominios con anticipación, cuando los datos

obligan a crear nuevas estructuras, el esquema puede ser actualizado en el transcurso de las operaciones para que el grafo siga actualizado.

La agilidad es un aspecto que se valora en la industria del software, existen varias metodologías que adoptan este concepto para desarrollar productos como los son *Scrum* y *Kanban*. Esta característica de las bases de datos orientadas a grafos se alinea a las prácticas modernas de la ingeniería de software, y de esta manera los grafos pueden evolucionar de forma natural junto con el resto de las aplicaciones que son construidas alrededor de ellos.

El trabajo realizado proveerá resultados sobre algunas de las BDG más populares en el mercado como son *GraphDB*, *JanusGraph*, *Neo4j*, y *TigerGraph*. El estudio está basado en datos duros en aspectos tales como soporte a diferentes tipos de conjuntos de datos, modelos en los cuales basan su implementación, lenguajes de consultas y soporte a algoritmos, tipos de aplicaciones, ambientes y configuraciones similares, para así ofrecerle al usuario una comparativa que permita ahorrar tiempo y dinero al elegir la mejor herramienta. Será un sistema que facilitará la elección de BDG tal como *Trip Advisor* ayuda al usuario al elegir elementos de un viaje a su conveniencia.

1.3 Problema

A pesar de que existen investigaciones sobre el desempeño de distintas herramientas BDG [2]- [3], hasta el momento no existe un *benchmarking* o investigaciones que estén hechas bajo citerios homogéneos y evaluen de forma equitativa.

A medida que las investigaciones e implementaciones sobre el uso de grafos han avanzado, la cantidad de datos a analizar y procesar también han incrementado. Dichos conjuntos de datos, en ocasiones referidos como "*Big Data*" son consumidos por tecnologías de procesamiento de datos masivos, y a pesar de que los recursos computacionales son cada día mayores, es importante saber cuál implementación es la más eficiente para la resolución de problemas específicos.

Muchas de las investigaciones no hacen públicos los métodos de medición, están orientados a inversionistas y vendedores grandes de tecnologías de la información. Y en su gran mayoría buscan un beneficio económico al publicar sus listas lo cual no es muy bien recibido ya que pueden llegar a causar dudas sobre la legitimidad de los resultados.

1.4 Objetivos

1.4.1 Objetivo General

Desarrollar una metodología para realizar un análisis comparativo de bases de datos basadas en grafos. La metodología debe contemplar que se utilicen los mismos conjuntos de datos de entrada, empleen los mismos instrumentos de medición, y se lleven a cabo en plataformas de prueba idénticas. Esto con la intención de que los resultados se puedan utilizar como punto de referencia académico y no como argumento de venta.

1.4.2 Objetivos Específicos

- Aplicar la metodología desarrollada en al menos tres bases de datos que estén siendo mantenidas de forma constante y que tengan casos de éxito en el mercado.

- Publicar al menos un artículo académico en revista o conferencia especializada en tecnología con orientación a sistemas de base de datos.
- Generar material de referencia para que otros proyectos o trabajos de obtención de grado puedan usar como base esta investigación en caso de que estén orientados al uso de grafos.
- Utilizar el conocimiento adquirido para proponer sistemas de evaluación de herramientas comunes dentro de las empresas y ayudar a la toma de decisiones cuando se necesite evaluarlas.
- Aprender lenguajes de manipulación de grafos para implementar algoritmos eficientes en el análisis de grafos.

1.5 Novedad científica, tecnológica o aportación

La plataforma y los resultados a obtener harán uso de las siguientes ramas de las ciencias computacionales:

- Procesamiento de datos masivos.
- Ejecución de algoritmos distribuidos.
- Ambientes de contenedores para aislar procesos
- Librerías de validación automática.

Los resultados a obtener en su mayoría serán almacenados en lenguajes de procesamiento computacional como JSON, CSV, o similares los cuales puedan ser importados a diferentes herramientas como base de datos, generadores de gráficas y visualización de resultados en plataformas web.

Un análisis de resultados justo y sin fines de convencer al mercado meta de usar algún sistema en específico. Se pretende proveer de una herramienta que ayude al consumidor a elegir un motor de BDG que sea el que mejor satisfaga las necesidades de los problemas a resolver.

Utilizando dichos datos de forma correcta harán que los tiempos de desarrollo e investigación se reduzcan y poder enfocar al equipo en implementar la solución deseada y no en la evaluación de herramientas. Gracias a esto, se podrían evitar situaciones que puedan llevar al fracaso una investigación o incluso a una empresa con recursos limitados.

Elegir herramientas puede causar fricciones entre el equipo de trabajo, generar costos al reemplazar una mala elección, causar una mala impresión pública por una elección de BDG incorrecta que no solucione el problema raíz. Al apoyar esta investigación se ayuda a cientos de futuros proyectos e investigaciones científicas que pueden fracasar desde el primer día al no tener una forma de elegir la BDG ideal para su problema a resolver.

2 ESTADO DEL ARTE O DE LA TÉCNICA

Diferentes publicaciones y estudios conforman las maneras actuales en las que se pueden comprender el desempeño de bases de datos para procesamiento de información masiva, incluyendo las bases de datos orientadas al manejo de grafos.

2.1 Comparaciones de aplicaciones de bases de datos orientadas a grafos.

A continuación, se exponen algunos de los estudios y análisis que se han hecho para comparar bases de datos orientadas a grafos.

2.1.1 Administración y Análisis de grafos con *Big Data*

El uso de grafos para administrar y analizar información a gran escala se ha popularizado a través de los últimos 20 años [4]. La necesidad de conocer el desempeño de las herramientas disponibles ha desatado una serie de estudios y publicaciones.

Se hacen comparaciones funcionales sobre los mecanismos de almacenamiento que comparan o describen características de BDGs contra otras bases de datos denominadas NoSQL [5] las cuales no son necesariamente hechas para representar relaciones logradas a través de la teoría de grafos.

De la misma manera describen usos y aplicaciones como el análisis de datos a través de redes que van desde los análisis cuantitativos [6], estudios sociológicos [7] y en el análisis de redes sociales en línea [8].

En la publicación de *Journal of Grid Computing* [9] se hizo un estudio donde categorizan algunas plataformas para el procesamiento de grafos. Se hace una clasificación por "familias" de productos la cual resulta interesante ya que se pueden identificar el origen de cada una de esas plataformas, dicha clasificación puede observarse en la Figura 2-1 Clasificación de Plataformas para Procesamiento de Grafos . La premisa de dicho análisis está enfocada a como los modelos de *Map Reduce* son ineficientes en términos de ancho de banda de red y uso de procesadores y como cada familia intenta solucionar dichos problemas de rendimiento a través de abstracciones programáticas para realizar análisis paralelos iterativos en grafos grandes.

Los datos capturados por cada familia son más descriptivos que comparativos:

- A la familia *Pregel* que introduce *Google* al mercado la clasifican como el primer sistema *Bulk Synchronous Parallel* (BSP) [10], modelo descrito en Figura 2-2 Modelo de Programación BSP ., el cual surge de una comparación con lo que se observa en el mundo de la computación secuencial dodne el modelo Von Neumman ha sido el de facto; En BSP, un sistema paralelo es visto como un conjunto de procesadores con memoria local e interconectados a través de una red de comunicación de topología transparente al usuario, dicha abstracción se aplica tanto al sistena

de memoria distribuida como de memoria compartida. En este trabajo, de forma posterior describen como funciona el paralelismo bajo su API nativa y concluyen con el origen, implementaciones y relaciones de cada descendiente de Pregel.

- De forma similar, la familia *GraphLab* es descrita como un Proyecto de código libre, para grafos a grande escala, y denota como mayor diferencia que esta familia se basa en una atracción de memoria compartida y el modelo de procesamiento GAS (*Gather, Apply, Scatter*). El cual se asemeja a BSP, pero tiene diferencias fundamentales a las empleadas por BSP. Finaliza el análisis con el mismo tipo de documentación que hizo en la familia *Pregel* para cada descendiente.
- La tercera familia categorizada como "Otros Sistemas" son aquellos que no comparten ninguno de los modelos de programación mencionados como GAS o BSP. Para cada uno de ellos destacan cualidades que están orientadas a la optimización de memoria y comunicaciones. Algunos de estos productos utilizan técnicas de partición de grafos a través de servidores en la nube, o introducen APIs que pueden ser aplicados a datos no estructurados y estructurados representados como tablas.

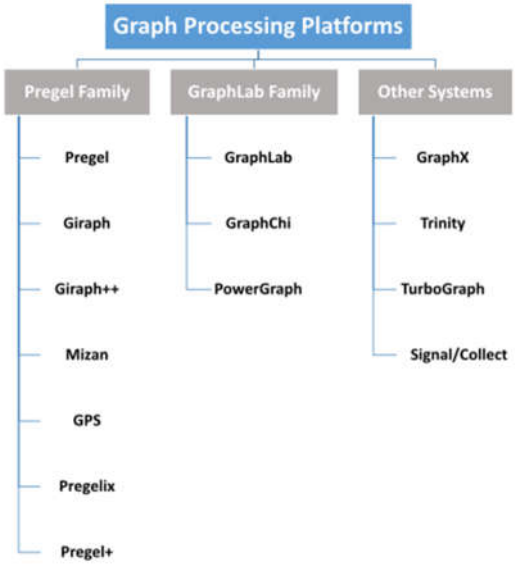

Figura 2-1 Clasificación de Plataformas para Procesamiento de Grafos [9].

Como parte de las conclusiones se hace énfasis en que tecnologías como *MapReduce*, considerada como una solución que puede manejar cualquier tipo de procesamiento de problemas basado en *Big Data*, dicha plataforma ya ha mostrado ciertas limitaciones en capacidad de computo de análisis en grafos de escala grande. Dichas limitaciones han generado una nueva generación de sistemas, las cuales son la base de nuestro estudio, denominadas Bases de Datos Orientadas a Grafos. Esta nueva generación está siendo denominada como *Big Data 2.0* y es por eso que nos interesa tener una plataforma comparativa y de evaluación para dichas BDG.

Otro tipo de comparaciones con sistemas de base de datos cien por ciento orientados a grafos han logrado obtener datos técnicos un poco más cercanos a lo que se desea obtener durante el desarrollo de este trabajo.

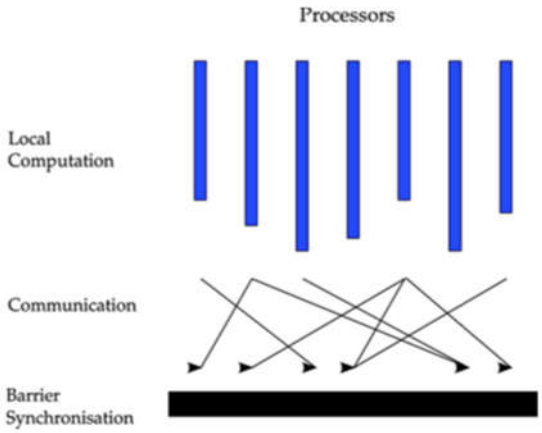

Figura 2-2 Modelo de Programación BSP [9].

A principios de 2017 se desarrolló una comparación donde la clasificación no es precisamente un árbol familiar sino la clasificación de BDG por tipo, dichas categorías son "Base de Datos de Grafos", "Procesamiento de Grafos" y "Sistemas de flujo de datos en Grafos".

- El desarrollo de la categoría "Base de Datos de Grafos", da un preámbulo al uso de herramientas desde 1970 para así abordar a los sistemas más recientes. Provee detalle del modelo de datos que usan y los lenguajes de consulta que soportan.
- El "Procesamiento de Grafos", se enfoca a los sistemas que funcionan de forma distribuida, tales como *Pregel* (una de las familias analizadas anteriormente). Denota la arquitectura de modelos de datos basados en *vertex-centric* y sus variaciones. Analiza el procesamiento desde el punto de vista de un vértice o de un grafo entero.
- Los "Sistemas de flujo de datos en Grafos" se asemejan a los discutido en el punto anterior, pero al intentar resolver problemas de complejidad analítica mayor, requieren de crear estructuras de grafos combinadas basadas en datos estructurados y no estructurados que se obtienen de diferentes fuentes, o incluso de combinar algoritmos para grafos junto con algoritmos que no necesariamente son orientados a grafos.

	Graph database systems	Graph processing systems	Graph dataflow systems	
Examples	Neo4j, Marklogic	Pregel, Giraph	Gelly, GraphX	Gradoop
Data model	PGM/RDF	Generic graph	Generic graph, Datasets	EPGM, Datasets
Graph collections	No	No	No	Yes
Query approach	Query languages	Vertex-/Graph-centric computation models	Vertex-centric computation, dataflow programs	
Scope	OLTP/Queries	Analytics	Analytics	Analytics
Scalability	Up/(Out)	Out	Out	Out
Persistency	Yes	No	No	Yes
Transactions	Yes	No	No	No
Graph visualization	Interactive traversal	No	No	No

Figura 2-3 Clasificación de BDGs en base a su capacidad y características [11]

En la comparación de características bajo diferentes enfoques de administración y análisis de datos en grafos de la Figura 2-3 Clasificación de BDGs en base a su capacidad y características, dicho enfoque sigue siendo comparativo y no de evaluación o medición de rendimiento bajo condiciones similares. De hecho, dentro de las conclusiones y retos a futuro incluyen la necesidad de tener un mecanismo de evaluación y puntos de referencia para poder elegir la mejor en base a las necesidades del usuario y que estas mismas evaluaciones sirvan para identificar cuellos de botella actuales que puedan ayudar a la siguiente generación de BDG e incluso para mejorar las existentes.

2.1.2 Evaluación y clasificación de bases de datos orientadas a procesamiento de grafos.

Existen estudios aislados que comparan el desempeño de diferentes bases de datos y sistemas de procesamiento de grafos. El problema con ellos es que los resultados no son comparables al usar diferentes sistemas de bases de datos para grafos, conjuntos de datos, aplicación de algoritmos o consultas, y entornos de pruebas que no son homogéneos. A continuación, un resumen y alcance de algunos de estos estudios. El resumén de cada trabajo es accesible en este documento dentro del Apéndice A.

2.1.2.1 An evaluation study of big data frameworks for graph processing

El artículo realiza la evaluación de plataformas *Big Data* (*Map-Reduce, Stratosphere, Hama, Giraph y GraphLab*) [2]. En esta publicación se hace una comparación de bases de datos orientada a grafos y otras que utilizan el esquema *MapReduce*. El algoritmo de prueba utilizado es *K-Core* en una implementación

distribuida para calcular la centralidad de un nodo. El ambiente de pruebas, un clúster *Amazon Web Services* (AWS) corriendo la versión 1.0.3 de *Apache Hadoop*. La comparación es hecha bajo las mismas condiciones de pruebas (hardware) y su escala fue de un máximo de 32 nodos. Para fines de nuestro trabajo es útil imitar condiciones de uso de hardware donde no ha beneficio para ningún sistema. El estudio está limitado a solo un algoritmo para el análisis de grafos, dejando a un lado otros que pudieran arrojar diferentes resultados por el tipo de operaciones involucrados. Su enfoque es diferenciar el desempeño entre plataformas orientadas a grafos y las demás que son para análisis de datos a grande escala.

Las conclusiones del trabajo son que todas las plataformas son mejores en tiempo de ejecución que *Hadoop* o versiones equivalentes de *MapReduce*. Recomiendan el uso de sistemas inspirados en *Pregel* cuando se tenga que solucionar un problema de grafos debido a su naturaleza *vertex-centric.* La **Error! Reference source not found.** nos provee una visión de los elementos evaluados en el estudio.

2.1.2.2 How well do graph-processing platforms perform? An empirical performance evaluation and analysis

Comparación y evaluación de 6 plataformas *Big Data* (*Hadoop, YARN, Stratosphere, Giraph y Neo4j*) [11]. El planteamiento que utilizan a la hora de describir el proceso de selección de algoritmos, sets de datos, entorno de validación y características a comparar es muy claro. El vehículo de ejecución de pruebas es la súper computadora DAS-4, lo cual tiene una ventaja al no estar usando máquinas virtuales y aprovechar los recursos físicos del sistema. Los resultados comparativos muestran gráficamente los datos de desempeño y se dan explicaciones breves del porque creen que se dieron cada uno de los resultados. No declaran a una plataforma ganadora, quizás no sea necesario ya que no buscaban encontrar a la mejor sino exponer fortalezas y debilidades de cada una.

Los resultados no son conclusos en cuanto al definir un ganador dentro de los tiempos de ejecución, pero todos *Hadoop* es el peor de todos. Lo cual parece ser obvio ya que es una plataforma que no está orientada al procesamiento de grafos. Ninguna de las plataformas pudo manejar todos los *datasets* al mismo tiempo. El nodo central utiliza pocos recursos en todos los casos, mientras que los nodos de computo varían el uso de los recursos entre todas las plataformas. En cuanto escalabilidad, algunas pueden hacerlo con el tamaño del clúster (horizontal), o en su número de procesadores (vertical). Incrementar el número de procesadores suele afectar el desempeño, sobre todo en grafos pequeños. La ingesta de datos en *Neo4j* se acerca a las características del grafo.

2.1.2.3 An experimental comparison of pregel-like graph processing systems.

Evaluación de sistemas tipo *Pregel*. Comparan *Giraph, GPS, Mizan* y *GraphLab* [12]. Una de las características que difiere de la mayoría de los enfoques descritos en este documento es que su comparación es orientada a ambientes de procesamiento de grafos y no a las bases de datos orientadas a grafos. La primera evalúa el procesamiento hecho en memoria para hacer cálculos mientras que las bases de datos son operaciones y administración de datos propios de cada grafo. Las plataformas fueron estresadas con cuatro diferentes categorías de algoritmos: *Random Walk*, *Sequential Traversal*, *Parallel Traversal* y *Graph Mutation*. Se seleccionaron conjuntos de datos pregenerados, pero todos extraídos de

datos del mundo real. Las métricas de desempeño a analizar son tiempo que requiere la plataforma para ser configurada, uso de tiempo de cómputo (en cada núcleo de los procesadores), uso de memoria, y uso de ancho de banda de red. Sus pruebas tienen como elemento central el uso de programación en paralelo para explotar las capacidades de cada plataforma. Sus conclusiones son específicas a los puntos donde cada base de datos se comporta mejor sobre las demás. La forma en que presentan los resultados de forma gráfica y a través de tablas facilita la lectura rápida y en caso de necesitarlo los detalles se puede navegar con facilidad cada uno de los algoritmos empleados y el porqué de su calificación en cada área.

La conclusión del experimento establece que el modo síncrono de *Giraph* y *GraphLab* tienen un desempeño general bueno, mientras que GPS sobresale en la eficiencia del uso de memoria. *Mizan* es el menos eficiente en el modo síncrono. En la mutación de grafos *Giraph* ofrece mejor opción que la utilizada por *Giraph*. La escalabilidad y el desempeño de *GraphLab* en su modo asíncrono es pobre y lleva a problema de comunicación.

Las principales áreas de mejora para cada plataforma:

- *Giraph*: Mejorar el balanceo de carga de trabajo para hacer más eficiente el uso de memoria.
- *GPS*: Tomar ventaja de la localidad de datos para mejorar la escalabilidad en el tiempo de configuración.
- *Mizan*: Añadir optimizaciones de procesamiento de mensajes para mejorar el desempeño y escalabilidad.
- *GraphLab*: Reducir el sobreuso de comunicación en su modo asíncrono.

2.1.2.4 Evaluation and analysis of distributed graph-parallel processing frameworks.

Aquí se presenta un trabajo que tiene como objetivo principal la evaluación de plataformas distribuidas, la mayoría orientadas a grafos como *PowerGraph, Giraph* y *GPS* [13]; casi todas las plataformas analizadas son del tipo *Pregel*, aunque incluyen también la plataforma Spark la cual no es específicamente orientada a grafos. Buscan resaltar las fortalezas y debilidades de cada plataforma. Es uno de los pocos de los estudios que proveen una sección preliminar sobre teoría de grafos. La teoría y fundamentos matemáticos es un tipo de información que suele obviarse en los estudios y aunque sea muy básica se aprecia una mayor formalidad al proporcionar dicha información para usuarios de BDG que no tengan estos fundamentos presentes. Los resultados gráficos son interesantes pero su acomodo en el documento los vuelve difícil de interpretar. Otra característica valiosa de la evaluación es que presentan cada sistema con un breve resumen ejecutivo donde dan contexto sobre su implementación, y describen ventajas y desventajas de cada plataforma. Las métricas están clasificadas en medición de procesamiento de datos, escalabilidad (un dato valioso ya que suele ser un punto común a la hora de tomar decisiones sobre la plataforma a elegir cuando empiezas a trabajar con grafos) y utilización de recursos. Los conjuntos de datos son extraídos de redes sociales reales con grafos dirigidos y no dirigidos. Los algoritmos usados son *PageRank*, *Shortest Path* y *Triangle Count*.

La conclusión de la evaluación dice que los sistemas orientados a grafos son mejores que los que se enfocan al procesamiento en paralelo. En caso de grafos grandes, *Spark* usa el menor tiempo en cargar los datos que los demás. La escalabilidad tiende a mejorar cuando los grafos analizados son más grandes. La

implementación mejorada de C++ para *PowerGraph* muestra mejor procesamiento que cualquier otra base de datos, sin embargo, su propuesta de computo asíncrono no muestra ninguna mejora significativa en el desempeño del experimento. *Giraph* es el mejor procesado cuando se trata de escalar, esto es debido a que no hay exceso de comunicación cuando más máquinas son utilizadas.

2.1.2.5 A performance evaluation of open source graph databases

Este trabajo esta dirigido a la evaluación de sistemas orientadas a grafos con código abierto [14]. La comparativa de este estudio utiliza plataformas que no son únicamente orientadas al manejo de grafos, mezclan sistemas SQL tradicionales, con sistemas NoSQL orientadas a grafos. Una de las mayores ventajas es que cada base de datos a comparar cuenta con al menos una versión *open source* o comunitaria lo cual suele ser atractivo para la academia o la industria de empresas nueva o con presupuesto limitado. El estudio es concreto y no agrega mucho contenido que no sea orientado a la comparación, incluso dan un acceso rápido a una tabla con el resumen de todo el análisis para cada plataforma donde no destacan quien es mejor sino datos concretos de las plataformas y sus características. Introducen una comparativa con sistemas *in-memory* para el uso de memoria compartida de forma masiva. Distintos algoritmos para grafos como *Single Source Shortest Path* (SSSP), *Shiloac-Viskin* (SV) y *PageRank* (PR) son analizados dentro del estudio. Concuerdan con la mayoría de los estudios al concluir que las nuevas plataformas BDG son más eficientes que las tradicionales como SQL en el uso y manipulación de grafos.

Las recomendaciones de la evaluación sugieren seguir mejorando las estructuras de datos y lenguajes de consulta para que sean más conscientes de la naturaleza de los datos en grafos. La BDG ideal debería entender consultas analíticas, que vayan más allá de seleccionar su vecindario.

2.1.2.6 Navigating the maze of graph analytics frameworks using massive graph datasets

El principal componente del trabajo es el análisis de oportunidades de mejora para las plataformas orientadas a grafos *Native, GraphLab, CombBLAS, SociaLite, Galios* y *Giraph* [15]. Utilizan 4 tipos de algoritmos para grafo: *PageRank, Breadth First Search, Collaborative Filtering,* y *Triangle Count*. Se utilizan conjuntos de datos reales y otro tanto de datos sintéticos de diversas fuentes y tipos de relaciones, lo interesante es que los datos sintéticos juegan un papel importante ya que se tiene mejor control sobre qué tipo de relaciones hay dentro del ecosistema del grafo. Todos los sets son de al menos cientos de miles de vértices y millones de aristas. El hardware es real, es decir no utilizan máquinas virtuales, de nuevo dando una mejor perspectiva del desempeño nativo de las BDG. Los resultados son orientados a puntos de mejora para cada plataforma identificando sus cuellos de botella, y las diferencias entre un sistema multi-nodos contra un sistema corriendo en una sola instancia de hardware. Uno de los trabajos más fuertes del estudio son sus fundamentos matemáticos sobre cada experimento, puede jugar como factor a favor en caso de que dicha información sea para personas con necesidades científicas para elegir plataformas, de la misma forma puede convertir el estudio en algo poco atractivo si cuando lo que se busca es simplemente saber áreas donde cada base de datos es fuerte o débil. Dependiendo del tipo de plataforma a evaluar también trabajaron en optimizar los algoritmos para que funcionaran de mejor manera en su implementación nativa.

El artículo no arroja datos comparativos sino orientados a optimizaciones nativas y el procesamiento de cada plataforma.

2.1.2.7 *Large-scale distributed graph computing systems: an experimental evaluation*

El objetivo del trabajo es realizar una evaluación sobre fortalezas y debilidades de diversos sistemas orientados a grafos como *Pregel, Giraph, GraphLab, GraphX, Mizan, GPS, Giraph++, Pregelix, Pregel+* y *Blogel* [3]. Sus objetivos generales están divididos en evaluar desempeño bajo diferentes características de grafos grandes, frente a distintos algoritmos, optimización de algoritmos, y probar la escalabilidad variando el número de máquinas, procesadores y distintos tipos de grafos. Como varios de los estudios encontrados, proveen una breve sección de antecedentes y conceptos básicos sobre teoría de grafos. La sección de datos es explicada en forma breve denotando que todos son datos del mundo real que van desde sistemas carreteros hasta grafos de redes sociales como *Twitter*. El vehículo de los experimentos son 15 servidores *Intel Xeon*, cada uno con 48 GB en RAM, discos SATA, conexión Gigabit corriendo CentOS 6.5. Los algoritmos utilizados son *PageRank*, *Diameter Estimation*, *SSSP*, *Hashmin,* SV, *Bipartitie Maximal Matching* (BMM) y *Graph Coloring* (GC).

Los resultados de la evaluación dictaminan que ninguna plataforma muestra desempeño superior en todas las áreas. Pero definitivamente suelen ser mejores *Pregel+* y GPS que *Giraph* y *GraphLab*. En particular, esto es logrado ya que *Pregel+* utiliza una combinación de *mirroring*, mensajes combinados y técnicas de petición-respuesta, y GPS se beneficia de su técnica LALP (*Large adjacency-list partitioning*). El desempeño de *GraphLab* se ve afectado por su uso excedido de bloqueo al implementar particiones basadas en corte de vértices para su balanceo de carga. *Giraph* tiende a ser el de peor desempeño al no tener una técnica específica para manejar cargas de trabajo sesgadas. Por último, *Pregel+* se beneficia de su implementación en C++ la cual reduce el uso de memoria lo cual tiende ser de 2 a 3 veces más rápido que implementaciones en Java.

2.1.2.8 *Comparative Analysis of Relational and Graph Databases*

La comparativa realizada en este trabajo no es específicamente sobre sistemas BDG, ya que utilizan un ejemplo de ellas para compararlas contra los sistemas de bases de datos relacionales [16]. Los ejemplos utilizados para el estudio son *MySQL* y *Neo4j*. Dentro del estudio se detallan especificaciones de alto nivel como lo son el soporte que casa base de datos tiene, la seguridad que proveen en base a limitaciones y accesos por usuario, la flexibilidad que tienen para actualizar sus esquemas, y el desempeño en búsquedas.

Al concluir la comparativa, ambos sistemas tuvieron un desempeño aceptable, pero Neo4j tuvo siempre mejor tiempo de respuesta. La flexibilidad que proveen los esquemas NoSQL como los grafos también fue notable sobre los esquemas rígidos de las bases de datos relacionales.

2.1.2.9 *Type of NoSQL Databases and its comparison with Relational Databases*

El enfoque que tiene el estudio es realizar una comparativa de distintos sistemas NoSQL contra las bases

de datos relacionales [17]. No se realiza un estudio formal sobre desempeño de consultas, logra realizar una descripción de cada modelo de datos disponible en los modelos NoSQL, están categorizadas en cinco: Llave-valor, Orientadas a columnas, Almacenamiento de documentos, Bases de datos para grafos, y Orientadas a objetos.

Los resultados resaltan las ventajas de las bases de datos NoSQL sobre las relacionales van desde la escalabilidad y amplios modelos de datos, hasta aspectos de administración limitada y la rapidez de cada una de ellas. Las desventajas suelen presentarse al no tener u n lenguaje de consultas estándar, y la inmadurez por su reciente aparición en el mercado.

2.1.2.10 Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4j and OrientDB

El trabajo está orientado a comparar un conjunto de BDGs. Analizan desde su perspectiva las bases de datos orientadas a grafos más populares las cuales son *AllegroGraph*, *ArangoDB*, *InfiniteGraphm Neo4j*, y *OrientDB* [18]. Dedican una sección para recordar las ventajas de este tipo de sistemas sobre otros modelos de datos. Una vez descritas las ventajas, describen de forma breve a cada sistema con sus principales características. Las características evaluadas son las siguientes: Flexibilidad del esquema, Lenguaje de consultas, *Sharding*, Respaldos de información, Modelos múltiples, Arquitectura, Escalabilidad, y su uso en ambientes *cloud*. La metodología de evaluación consiste en dar una calificación del 0 al 4 en cada categoría a cada base de datos. Donde 0 significa que no hay soporte para dicha característica; 1 es para un soporte malo; 2 es para un soporte normal/promedio; 3 es soporte bueno; 4 es soporte excelente.

La conclusión de este estudio hace hincapié en que, aunque todas son BDGs y están diseñadas para el mismo propósito, el cual es el manejo de volúmenes grandes de información con alta conectividad, cada una de ellas provee diferentes funcionalidades. Por lo cual es importante tener en cuenta el tipo de problema a resolver, las plataformas que se quieren conectar a la solución y la interacción que puede tener con el resto del sistema a implementar. Dentro de los puntajes asignados por la metodología la sumatoria arroja estos resultados en orden de mayor a menor puntaje: *ArangoDB* (26 pts.), *Neo4j* (26 pts.), *OrientDB* (25pts.), *AllegroGraph* (23 pts.), *InfiniteGraph* (21 pts.).

2.1.2.11 Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark

El trabajo esta basado en hacer una evaluación de BDGs a las cuales se les aplica una versión del *benchmark HPC Scalable Graph Analysis Benchmark* [19]. El *benchmark* es un desarrollo que surge de la colaboración de investigadores de universidades e ingenieros de la industria para analizar los distintos aspectos del desempeño de una BDG. En la comparativa se analizan las cuatro bases de datos orientadas a grafos que a su perspectiva eran las más populares al momento de realizar su trabajo, estas son *Neo4j*, *Jena*, *HypergraphDB*, y *DEX*. El *benchmark* provee de un algoritmo para generar conjuntos de datos de diferentes tamaños los cuales fueron utilizados para tener grafos de tres dimensiones, con un total de nodos de mil, treinta y dos mil, y 1 millón de nodos. Al agregar relaciones y ponderación en las relaciones se obtienen de 10 mil a más de 9 millones de objetos. Y se aplican 4 núcleos, cada uno de ellos está diseñado

para medir el desempeño de las bases de datos, el primero mide operaciones de inserción, el segundo está orientado a la búsqueda de nodos con mayor peso, el tercero aplica detección de comunidades, y el último hace recorridos completos del grafo.

Al finalizar, todas las bases de datos muestran un desempeño aceptable y cercano cuando el conjunto de datos es pequeño, al incrementar el tamaño de datos, y con la configuración de hardware que implementaron, solo Neo4j y DEX fueron capaces de cargar los datos. La carga de datos fue más rápida a través de DEX, Neo4j tuvo mejor desempeño en el tercer núcleo con la base de datos de mayor tamaño. El recorrido de grafos muestra que Neo4j tiene mejor desempeño cuando la base de datos es pequeña, pero sufre de una degradación importante cuando se escala a un millón de nodos; mientras que DEX logra un mejor alcance de escalabilidad en el mismo recorrido de grafos. Se concluye con que DEX tuvo mejores resultados sobre todo cuando la escalabilidad es aplicada, en los casos donde Neo4j tuvo mejores resultados, DEX muestra un desempeño muy cercano.

2.1.3 Publicaciones comerciales

El consumo de resultados sobre el desempeño de herramientas BDG y en general de sistemas que soportan el procesamiento de *Big Data* se han apoyado de publicaciones comerciales tal como las hechas por *Gartner Magic Quadrant* [20].

Las búsquedas en internet con el término "*Magic Quadrant*" arrojan cientos de resultados de diferentes empresas mostrando a los usuarios su posición más reciente dentro del cuadrante. A grandes rasgos para poder ser evaluado, un interesado debe responder una extensa documentación y contar con retroalimentación de al menos 20 clientes del interesado. Una vez ingresado se realiza un proceso interno y al final se publica una lista de resultados basado en un cuadrante el cual tiene las categorías de "nicho", "retador", "visionario y "líder".

Si bien *Gartner* advierte hacer uso de esta información para evaluar en distintas categorías las fortalezas y debilidades de los sistemas, la firma dice que está designado para reducir la búsqueda de proveedores. Pero esto genera que haya proveedores que no sean considerados al no formar parte del cuadrante de *Gartner*.

La legitimidad de dichos estudios causa controversia al no tener publicada de forma abierta la metodología de sus evaluaciones y al tener ya diversas demandas por sus publicaciones consideradas como injustas y favorecedoras a unos pocos proveedores [21].

En las publicaciones de *Big Data* se habla sobre la comparativa de tipos de sistemas orientados a grafos [1], uno de los aspectos notados es que dicho estudio dedica un capítulo entero a la herramienta *Gradoop* [22], el cual es desarrollado por la gente involucrada en dicho análisis de bases de datos.

2.1.4 Grafos con estructura no orientada a *Big Data*

Ya se ha descrito la imporancia de conocer el desempeño de las Bases de Datos orientadas a Grafos a escala, especialmente cuando los datos incluyen millones de nodos relacionados, pero hay casos en el que los usuarios buscan otro tipo de características, entre ellas se encuentran aquellas donde se pueda manipular de forma eficiente las relaciones entre uno o varios grafos sin necesidad de que los datos sean considerablemente grandes o cumplir las características para ser considerados *Big Data*. Para dichos casos no se encontraron estudios formales como los presentados en las secciones previas.

2.2 Benchmarks orientados a grafos

El cómputo para el procesamiento de grafos se ha vuelto relevante dentro de la última década, es por eso que distintas organizaciones han desarrollado *benchmarks* que puedan medir el desempeño de sistemas computacionales al correr algoritmos que tengan como objetivo resolver problemas del mundo real con grandes cantidades de datos, representados como un conjunto de nodos y sus conexiones.

2.2.1 Graph500 Benchmark

Inspirados en la lista *Top500* la cual clasifica a las 500 supercomputadoras más poderosas del mundo basada en su capacidad de procesar operaciones de punto flotante por segundo, el grupo de *Graph500* publicó en 2010 una introducción a su *benchmark* orientado al procesamiento de información masiva a través de su representación en grafos [23].

Las propiedades del *benchmark* están hechas para exhibir elementos tales como el núcleo fundamental con alcance de aplicaciones amplio, asociar los elementos a problemas y conjuntos de datos del mundo real.

El conjunto de datos descritos en la Tabla 2-1 **Algoritmos, datos, e implementaciones del *Benchmark* Graph500**, son autogenerados por el *benchmark,* pero están basados en datos reales obtenidos de diferentes fuentes, el mecanismo utilizado para generarlos es usando una matriz de adyacencia con la cantidad de vértices y aristas que cada conjunto describe en su origen.

Tabla 2-1 Algoritmos, datos, e implementaciones del *Benchmark* Graph500

Algoritmos	Datos	Implementaciones
Búsqueda. BFS (Búsqueda en amplitud).	Ciberseguridad 15 billones de entradas	Memoria distribuida
Optimización. SSSP Camino más corto	Informática médica. 50 millones de patentes	*MapReduce.*
Orientado a aristas. Datos independientes que no son subconjunto de otros datos independientes.	Enriquecimiento de datos. Petabytes de información	Memoria compartida multi-hilos.
	Redes sociales. Sin límites	
	Redes simbólicas. Petabytes.	

Los algoritmos orientados a grafos fueron elegidos con el fin de ejercitar las relaciones entre vértices con implementaciones que estresan a los sistemas computacionales modernos. La búsqueda en amplitud y encontrar el camino más corto son ampliamente conocidos y analizados en otros estudios de desempeño de motores de bases de datos orientados a grafos.

El comité que desarrolla la metodología descrita permite a los distintos centros de cómputo y desarrolladores de arquitecturas de computadoras proveer implementaciones optimizadas para incorporarse a la lista de los mejores sistemas procesando información masiva.

El trabajo presentado en este documento hace énfasis en encontrar modelos de uso comunes en el procesamiento de información relacionada y representada como un grafo, de esta forma se encontrarán las fortalezas y puntos de mejora de las bases de datos seleccionadas. Le facilitará al lector la elección de un

motor y lenguaje que sea el adecuado para obtener los resultados deseados en su investigación y/o producto.

La base del estudio a realizar esta basado en conceptos de teoría de grafos y sus diferentes características que se comentaran en la siguiente sección.

3 MARCO TEÓRICO/CONCEPTUAL

En este capítulo se presentan las bases teóricas y conceptuales sobre bases de datos orientadas a grafos, sus aplicaciones y el tipo de recursos y algoritmos que se ejercen para su comparativa.

3.1 Enfoque a Bases de Datos basadas en Grafos

Los estudios recientes y similares a la descripción del trabajo en desarrollo han enfocado sus resultados en saber cómo las BDG se comportan contra plataformas que no son orientadas a grafos. Estos mismos denotan en su totalidad que las BDG son más eficientes que aquellas plataformas *Big Data* que no se especializan en análisis de grafos [2]- [13]. La Figura 3-1. Representación de un grafo en herramienta *Neo4j*, muestra un ejemplo del nivel de detalle que una BDG puede lograr con sus visualizaciones de estructuras de datos conectadas para analizar sus relaciones.

Figura 3-1. Representación de un grafo en herramienta *Neo4j* [24]

3.1.1 Introducción a la teoría de grafos

Los grafos son una forma de representar redes o colección de objetos interconectados entre sí. En matemáticas, los grafos están definidos como pares ordenados de vértices y aristas [25].

Por definición, un grafo se puede representar de la siguiente forma G = (V, E). Donde V es un conjunto de nodos también llamados vértices, y E es un conjunto de aristas.

Las aristas sirven como conectores entre vértices pueden tener dirección, o no contar con el concepto de origen y destino. Cuando las aristas no tienen dirección, se dice que es un grafo no dirigido, en dichos casos el flujo entre nodos de puede dar en ambos sentidos. En el caso de que las aristas tengan una dirección, entonces el flujo en el grafo está limitado por dicha condición y se introduce el concepto de

vértices origen y destino. Ejemplo de grafos dirigidos y no dirigidos se describen en la Figura 3-2. Grafo Dirigido vs No Dirigido

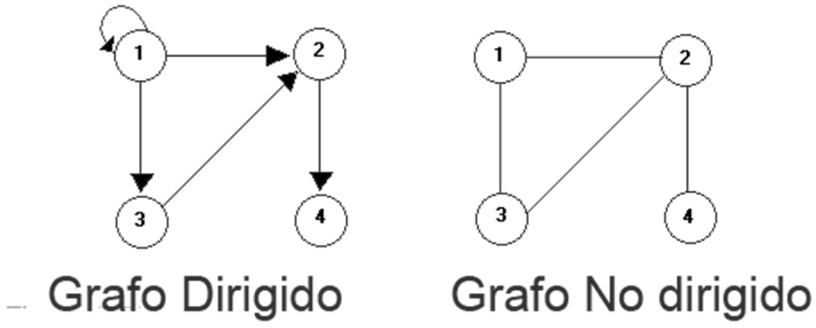

Figura 3-2. Grafo Dirigido vs No Dirigido [26]

3.1.2 Aplicaciones de la Teoría de Grafos

Los grafos en la vida real son utilizados en diversos ámbitos de la tecnología y ciencia:

- Conexiones entre amigos en redes sociales, donde cada usuario es un vértice, cuando los usuarios se conectan entonces están creando una arista.
- Utilización de geolocalización en servicios de mapas como *Google* y *Yahoo Maps*. Sirven por ejemplo para encontrar la ruta óptima ente dos o más puntos del mapa.
- La búsqueda de sitios web en diferentes motores de búsqueda. Las páginas de internet están conectadas entre ellas a través de vínculos, cada página es un vértice y el vínculo entre ellas funge como arista.
- En sitios de comercio electrónico, las recomendaciones a usuarios en base a sus gustos son logradas a través del uso de grafos y el consumo de otros usuarios con gustos similares.

Como se observa, cualquier red o conjunto de objetos conectados entre sí son representables a través de un grafo, es por eso que el papel de las BDG en la manipulación de dichos grafos se ha vuelto de suma importancia en las ciencias computacionales.

3.2 Elementos de estudio para grafos.

Los criterios relevantes del trabajo para analizar el desempeño de las BDG son los algoritmos que soportan de forma nativa y la capacidad de utilizar distintos conjuntos de datos para la evaluación y obtención de puntos de referencia. Existen una gran cantidad de grafos construidos basado en escenarios reales de redes sociales, estudios científicos, entre otras representaciones de objetos interconectados que sirven para ser utilizados al momento de estudiar grafos.

Los algoritmos utilizados de forma frecuente en la teoría de grafos tienen implementaciones óptimas en varios lenguajes de programación y están distribuidos en algunas versiones de las bases de datos.

3.2.1 Conjuntos de datos

Los estudios de plataformas de procesamiento de datos han hecho uso de conjuntos de datos publicados por diversas empresas y en muchas ocasiones consolidados por instituciones académicas que facilitan la selección del tipo de elementos a estudiar.

La universidad de *Stanford* (Estados Unidos de América) provee a través de su sitio SNAP [27] una librería de propósito general para el análisis de redes y minería de grafos. Las categorías en las cuales despliegan su información están basadas en el uso dentro de la vida real tales como redes sociales, redes de comunicación, comunidades, grafos web, mapas, productos en sitios de ventas, entre otros. Y después los puedes seleccionar con base a sus tipos (dirigidos o no dirigidos), cantidad de vértices y aristas, además de proveer una descripción general del grafo.

Otros archivos universitarios como *Konect* [28] de la universidad *Koblenz* (Alemania) e ICON [29] de la universidad de Colorado (Estados Unidos de América) permiten hacer selecciones en base al dominio de estudio o al hacer abstracciones de sitios web como *Amazon* y *Wikipedia* para representar dichos sitios en forma de grafos.

3.2.2 Algoritmos orientados a grafos

Los algoritmos orientados a grafos son técnicas, instrucciones y reglas definidas para hacer búsquedas, operaciones de cómputo y manipulación de elementos.

Las operaciones de búsqueda más comunes son las de búsqueda en amplitud y profundidad BFS y DFS por sus siglas en inglés. De ahí se derivan algoritmos que son útiles para encontrar la distancia más corta entre 2 vértices, o todos los caminos posibles entre un par de vértices.

Todas las BDG a ser analizadas proveen de al menos una implementación de las siguientes categorías de algoritmos para poder ser comparadas entre sí:

- Algoritmos de centralidad como *Page Rank*.
- Detección de comunidades como *Weakly Connected Components*.
- Descubrimiento de caminos como *K-Hop Path*.
- Árbol de recubrimiento mínimo como *Prim* y *Kruskal*

En las siguientes secciones se proveen detalles de los algoritmos utilizados frecuentemente para evaluar bases de datos orientadas a grafos.

3.2.2.1 K-Hop-Path

El algoritmo *k-hop-path* o también conocido como *K-Neighborhood* hace un conteo del número de caminos que se pueden formar de longitud k desde un nodo inicial, siendo k un número seleccionado por el usuario. Se utiliza generalmente para conocer el desempeño del recorrido de grafos. La metodología propone aplicar el algoritmo sobre la siguiente configuración:

- Preseleccionar 300 nodos y aplicar el algoritmo para k igual a 1 y 2.
- Preseleccionar 10 nodos y aplicar el algoritmo para k igual a 3 y 6.

La preselección consiste en hacer una elección de N nodos para cada conjunto de datos y guardarlos para ser utilizados en todas las bases de datos, la mayoría de los lenguajes de programación modernos permiten hacer esta selección de nodos a través del uso de una “semilla” para fines de reproducir experimentos como el que haremos más adelante con el caso de estudio aplicado. Esta parte es indispensable ya que, de no hacerse, entonces el algoritmo estaría siendo aplicado a diferentes nodos lo cual invalidaría resultados para fines comparativos. El recorrido aplicado en las consultas inicia de cada uno de los nodos

preseleccionados. Y va a encontrar todas las conexiones (caminos) que estén a una distancia k. Las consultas (una por cada nodo preseleccionado), son ejecutadas de forma secuencial. Cada nodo inicial tiene un tamaño diferente para su "vecindario", por lo cual el algoritmo calcula el tamaño promedio, este dato es el que se captura para ser comparado posteriormente. El enfoque del tamaño promedio nos ayuda a en minimizar la latencia en la red al ignorar otros datos como la lista completa de nodos que forman dicho "vecindario". Una buena práctica es validar que el promedio de tamaños de las comunidades sea verificado entre todas las bases de datos, la teoría dice que este dato debe ser idéntico entre todas las bases de datos evaluadas. De forma adicional, los usuarios pueden configurar o definir un tiempo de expiración, en donde si es cumplido antes de que el algoritmo termine su ejecución, entonces la prueba es marcada como fallada o incapaz de ser completada en un tiempo específico. El tiempo fuera puede servir como una referencia importante para descartar el uso de una base de datos para grafos.

3.2.2.2 Weakly Connected Components (WCC)

WCC es un algoritmo que detecta conjuntos de vértices y aristas conectados que pueden ser alcanzados entre ellos, en este caso la dirección de las aristas es ignorados. El algoritmo WCC hace un recorrido total del grafo, toca todos los nodos y aristas al menos una ocasión.

3.2.2.3 PageRank (PR)

PR es un algoritmo iterativo el cual recorre cada arista en cada una de sus iteraciones y calcula una calificación para cada uno de los nodos. El algoritmo termina después de varias iteraciones o ya que la calificación para los nodos converge en iteraciones consecutivas.

Un número de iteraciones recomendado para el algoritmo es 10. Pero cualquier número definido debe ser utilizado para la evaluación de todas las bases de datos. Similar a WCC, para este algoritmo se recorre todo el grafo para obtener el resultado de todos los nodos.

3.3 Reducción de elementos externos a través del uso de contenedores

La tecnología de contenedores es utilizada en la industria para hacer despliegue de software en servidores e infraestructuras basadas en la nube. Esto facilita el desarrollo de aplicaciones y permite que la migración de dichas aplicaciones sea compatible a través de diferentes plataformas.

Entre las aplicaciones de contenedores utilizadas con mayor frecuencia se encuentran *Docker* y *Rkt* (*Rocket*). La arquitectura básica de un contendor con la tecnología *Docker* puede ser observada en la Figura 3-3. Abstracción de un contendor *Docker*.

Dentro del desarrollo del TOG, son un par las características que estamos interesados en explorar y aplicar a la hora de hacer la evaluación de BDGs: Facilidad de ejecución en diversos sistemas y arquitecturas de hardware; y aislamiento de procesos dentro de su entorno de ejecución.

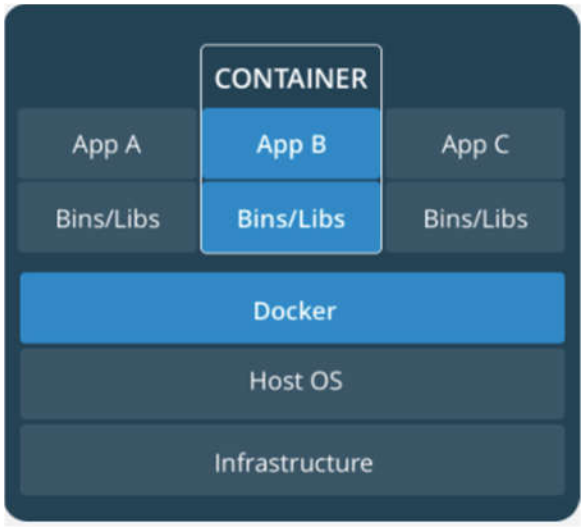

Figura 3-3. Abstracción de un contendor *Docker* [30]

3.3.1 Independencia de configuración de sistema

Al no tener aun definida una plataforma o conjunto de sistemas meta para hacer las comparaciones, se explora el uso de contenedores para crear imágenes con las bases de datos previamente configuradas. De esta forma no se necesitará hacer trabajo en la plataforma elegida y de igual forma el set de pruebas puede ser portado con esfuerzo mínimo a diferentes sistemas. Un gran conjunto de aplicaciones ha optado por reproducir sus instalaciones y configuraciones a través de contendores, la Figura 3-4. Portabilidad de contenedores muestra una forma en la que un servidor puede ser provisionado con los servicios necesarios a través de ellos.

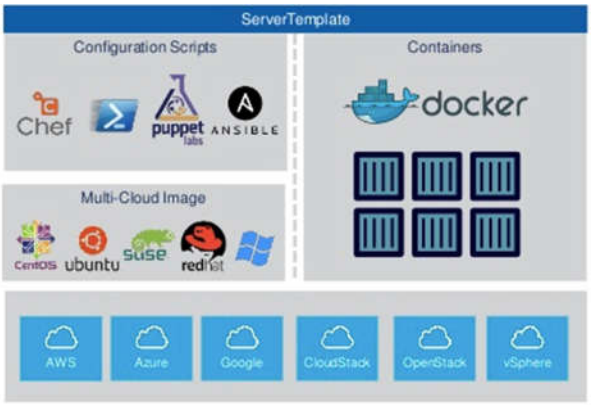

Figura 3-4. Portabilidad de contenedores [31]

3.3.2 Aislamiento de procesos

Al querer evaluar el uso de recursos del sistema tales como el uso de CPU y memoria de las plataformas, al lanzar las pruebas dentro de ambientes aislados se puede hacer un monitoreo efectivo del uso total de recursos del sistema para una BDG en específico. En la Figura 3-5. Contenedores aislados bajo un mismo *kernel* se detalla como una instancia de *kernel* poder servir a diferentes instancias de sistemas operativos como *Debian*, *Ubuntu*, y *Alpine* para poder servir aplicaciones como *Tomcat*, *MySQL* y librerías estáticas.

La utilización de recursos del sistema se pueden obtener de forma nativa a través de los implementadores

de contenedores como *Docker*, o incluso utilizar librerías externas especializadas en el uso de recurso dentro de un contenedor.

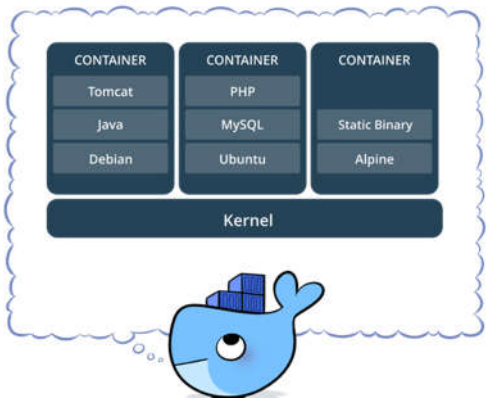

Figura 3-5. Contenedores aislados bajo un mismo *kernel* [30]

Parte del contenido deseado se logra utilizando los conceptos base de la teoría matemática para la creación y manipulación de datos conectados, y tecnologías en sistemas operativos que permiten aislar procesos y la utilización de recursos. También se utilizarán metodologías y técnicas existentes junto con el desarrollo de una plataforma que permita comparar los resultados de cada ejecución, en la siguiente sección se describe la estrategia completa.

4 DESARROLLO METODOLÓGICO

El desarrollo de la evaluación y *benchmark* de BDGs comprendidos en este trabajo está acotado a cuatro productos los cuales se consideran modernos y en constante evolución para mejorar su rendimiento. El criterio de selección está basado en el sitio web *DB Engines* [32]. La popularidad de las bases de datos según este sitio está basada en el número de veces que las bases de datos son mencionadas en buscadores como *Google* y *Bing*, incluyen datos de estadísticas generadas por *Google Trends*, menciones en foros de discusiones técnicas como *Stack Overflow*, ofertas de trabajos que incluyen dichas tecnologías en *Indeed*, menciones de los sistemas en perfiles públicos de profesionistas en *LinkedIn*, y relevancia de menciones en redes sociales como *Twitter*. En base a esos resultados, se encontró que las siguientes bases de datos orientadas a grafos son actualmente las que queremos comparar para obtener resultados y analizar su desempeño en distintas áreas:

- *GraphDB* v8.6.1 *Free Edition*. Desarrollado por *Ontotext*
- *JanusGraph* v0.2.1 *Apache License*. Proyecto *Open Source.*
- *Neo4j* v3.4.7 *Community Edition*. Desarrollado pro *Neo4j Inc*.
- *TigerGraph* v2.1.4 *Developer Edition*. Desarrollado por *TigerGraph*.

La contribución más importante de este trabajo es el desarrollo de una metodología para comparar las bases de datos para grafos la cual será descrita en esta sección.

- Para cada una de las bases de datos que se analizan, se provee una guía sobre sus fortalezas y en qué áreas existen oportunidades para mejorar
- También se proporciona una breve introducción a los lenguajes que cada sistema utiliza para explorar los grafos.

Posteriormente se utiliza la metodología descrita para crear un caso de estudio con las bases de datos que mencionamos anteriormente.

Se construirán un conjunto de scripts que ayudan a la automatización para ejecutar las instancias de los servidores en *Amazon EC2* y crear la configuración mínima para instalar las diferentes bases de datos. La automatización en su segundo paso será hacer la instalación y configuración de las herramientas.

4.1 Levantamiento de requerimientos

El análisis de las bases de datos orientadas a grafos incluidas en este documento cumple con los criterios de selección, configuración y medición de elementos del sistema descritos en la Tabla 4-1. **Tabla de Requerimientos**.

Tabla 4-1. Tabla de Requerimientos para aplicar al *Benchmark* y Evaluación de Bases de Datos orientadas a Grafos

ID	Requerimientos
Selección de Bases de Datos orientadas a Grafos	
REQ-001	Los motores de bases de datos a seleccionar deben ser instalados desde imágenes oficiales del producto.
REQ-002	Las bases de datos que se analizarán como mínimo son *GraphDB, JanusGraph, Neo4j, y TigerGraph*
REQ-003	Se deben escoger las últimas versiones estables publicadas para cada uno de los motores BDG. La fecha de selección debe hacerse dentro del primer trimestre de 2018.
Ambientes de Validación	
REQ-004	El entorno de validación debe ser capaz de generar las condiciones mínimas (dependencias) necesarias para ejecutar un conjunto de pruebas sobre el motor de base de datos.
REQ-005	El entorno de validación debe de proveer de un archivo de configuración general para establecer el tipo de pruebas a correr, numero de iteraciones, tamaño de grafos, BDG involucrada, tiempo de ejecución.
REQ-006	El entorno de validación deber proveer métodos para propagar configuraciones específicas para cada motor de base de datos orientado a grafos.
REQ-007	Las comparaciones realizadas por el entorno de validación deben generarse con conjuntos de datos iguales, configuraciones y ambientes de ejecución semejantes.
REQ-008	El ambiente de validación debe poder generar condiciones donde se estresen la memoria para generar condiciones de fallo al estar ejecutando operaciones sobre la BDG.
Sets de Datos	
REQ-009	Los conjuntos de datos seleccionados para realizar las pruebas deben poderse utilizar en todos los motores BDG.
REQ-010	La selección de datos debe tener al menos un objetivo a medir los cuales pueden caer en alguna de las siguientes categorías: Consultas. Análisis. Desempeño, Escalabilidad, Uso de memoria y Transacciones.
REQ-011	Se debe procesar al menos un set de datos por cada criterio del trabajo (Consultas. Análisis. Desempeño, Escalabilidad, Uso de memoria y Transacciones) en cada BDG.
Recolección y Publicación de Resultados	
REQ-012	Al finalizar cada ejecución, se debe proveer de archivos de resultados que se puedan analizar y comparar en una interfaz gráfica entendible para el consumidor humano.
REQ-013	Al finalizar las ejecuciones se debe construir de forma consolidada un reporte general por cada BDG.

4.2 Descripción de la metodología

Proponemos una metodología que nos ayude a obtener resultados sobre el desempeño de una base de datos orientada a grafos y poder compararlas entre ellas. Consideramos que es importante ya que la creciente demanda por analizar datos altamente conectados ha creado una necesidad de crear productos nuevos orientados a este funcionamiento. Aplicar la metodología ayudara a los usuarios a saber si un producto nuevo o existente puede resolver o mejorar comportamientos de aplicaciones o investigaciones. Los pasos de esta metodología están representados con su flujo visual en la Figura 4-1 **Metodología para evaluar BDGs.**

La primera pregunta que generalmente se necesita resolver y documentar: ¿El sistema provee de alguna interfaz o herramienta para importar/cargar datos a escala? Es probable que todos los sistemas tengan una implementación para este paso, pero es importante considerar cuales son y que opciones proporcionan debido a las grandes cantidades de datos nuevos que se generan cada día, y como las aplicaciones pudieran

presentar problemas de escalabilidad y disponibilidad al ingresar información nueva. Dentro del mecanismo para probar las herramientas para cargar datos se miden un par de métricas, el tiempo que se consume para cargar un conjunto de datos, y el espacio que utiliza la información en el sistema de almacenamiento (discos duros). Si por alguna razón no se encuentra un mecanismo para carga de datos masivos, se debe considerar si dicha limitante afecta el uso del sistema para el problema a resolver, si es así, entonces se considera por descartada la base de datos y se recomienda no continuar con la evaluación.

Una vez que se termina con la captura de información sobre el desempeño en la carga de datos, pasamos a las siguientes preguntas: ¿Cuáles son los lenguajes de consulta que el sistema para grafos utiliza? ¿y cuál es el soporte que tienen para la ejecución de algoritmos para grafos? El soporte para algoritmos pudiera ser una pregunta obvia, pero no todos los lenguajes para consultas tienen esta funcionalidad para lograr consultas para relaciones complejas, o análisis sobre comunidades dentro de los grafos. Se proveerán ejemplos sobre esto cuando analicemos las capacidades de los lenguajes primarios de las bases de datos que aplicamos en el caso de uso dentro de este documento. Si ninguno de los lenguajes para consulta proporciona funcionalidad para operaciones básicas de un grafo, entonces la comparación es concluida y no procedemos con los siguientes pasos. En caso contrario, comenzamos a documentar características y detalles del lenguaje para poder ser comparados con los demás:

- ¿El lenguaje es declarativo o imperativo?
- Casos de uso principales que han sido resueltos en el mercado.
- Soporte de algoritmos por tipo:
    - Centralidades
    - Detección de comunidades
    - Exploración de caminos
    - Similaridades
    - Predicción de conexiones
    - Pre procesamiento
    - Árboles
- ¿Puede detectar nodos y vértices duplicados?
- ¿Puede detectar ciclos?
- ¿El lenguaje cumple con las normas para ser considerado *Turing* completo?
- ¿El lenguaje cumple para ser considerado SQL completo?

Una vez que esta información es capturada se puede proceder al siguiente paso. Es en este paso donde se considera que la automatización juega un papel importante, ya que con ella se pude lograr reproducibilidad de resultados, o lograr hacer actualizaciones de versiones sobre una misma herramienta con esfuerzos menores. Pero como cualquier proceso de automatización, se recomienda hacer una primera ejecución manual con la cual se pueda familiarizar con las funciones básicas del sistema, conocer las interfaces gráficas y por líneas de comando y recabar información sobre la tecnología que está siendo utilizada por cada motor de base de datos. Al recabar esta información se puede complementar pasos anteriores al documentar dichos elementos junto con el tipo de licencias con las cuales se puede trabajar, el soporte que la comunidad aporta a usuarios, fechas de lanzamientos y su versión más reciente, interfaces de programación, y así poder evaluar que tan rápido es poder

utilizar el sistema para las personas que están empezando a conocer la herramienta. Es importante documentar esa información para futuras referencias.

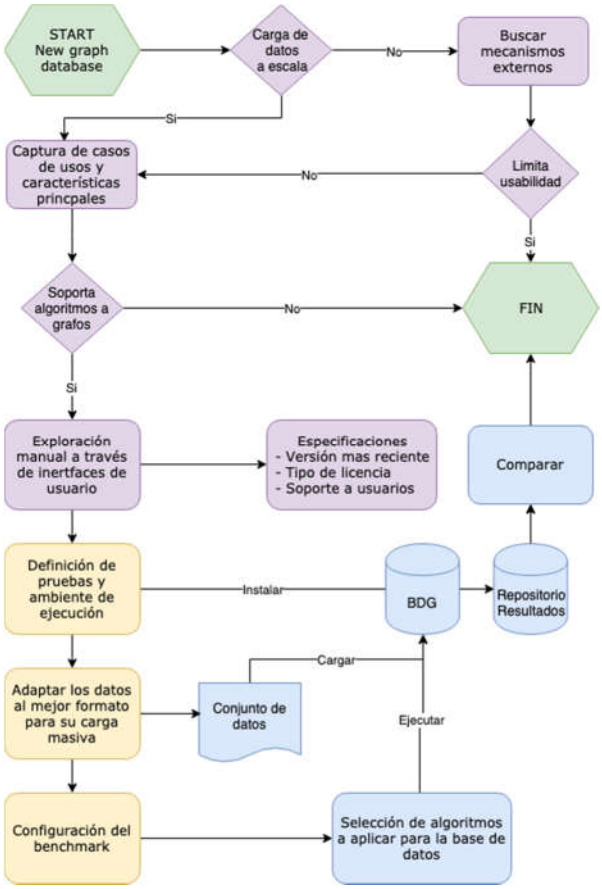

Figura 4-1 Metodología para evaluar BDGs

El proceso que se utiliza para obtener los datos de desempeño junto con sus requerimientos funcionales y no funcionales considera que cada uno de los sistemas sean instalados bajo el mismo tipo de infraestructura de hardware y software. El hardware incluye, pero no está limitado al mismo tipo de CPUs, tamaño de la memoria RAM, tipo y espacio de almacenamiento en disco duro, e interfaces de red; en cuanto al software aplica lo mismo para la versión del sistema operativo, configuración de BIOS, e instalar únicamente las librerías y dependencias indispensables para que la base de datos pueda operar en su mejor escala.

Una vez que la plataforma (entorno de ejecución) ha sido configurada y cumpla con los requisitos mínimos que requiere la base de datos para grafos, y tenga las mismas especificaciones en donde se han ejecutado los demás sistemas, se procede a realizar la instalación y configuración de la base de datos. Este paso puede ser automatizado en base a las lecciones aprendidas durante la exploración manual del sistema. Los pasos para la instalación y configuración deben ser obtenidos de las páginas y/o repositorios oficiales que proveen los desarrolladores y comunidad del sistema; las optimizaciones que sean recomendadas pueden ser aplicadas en base a las necesidades del estudio y que hayan sido recomendadas por parte de los foros oficiales para la exploración de datos masivos.

La configuración y ejecución del *benchmark* es el siguiente paso de la metodología. De forma similar a lo que hemos explicado sobre el entorno de ejecución, el *benchmark* consiste en cargar los mismos conjuntos de datos y tipo de algoritmos sobre la base de datos que está siendo evaluada. En la siguiente sección se provee una descripción detallada del proceso para correr el *benchmark*. Una vez ejecutado ese paso, la metodología concluye en la captura de resultados y realizar una comparación con las demás bases de datos para grafos en las áreas que proveen resultados similares. Depende del usuario final el enfoque y la profundidad de la comparativa, los resultados básicos son muy sencillos de interpretar y comparar.

4.3 Descripción del *benchmark*

El *benchmark* que se propone y reproduce en este trabajo está basado en los datos y automatización que proporciona el proyecto *ecosystem/bencmark* que mantiene el repositorio oficial de *TigerGraph* en *GitHub* [33]. El *benchmark* está compuesto de tres etapas. La configuración del *benchmark*, las pruebas de carga de datos, y las pruebas de desempeño de consultas.

4.3.1 Configuración del *benchmak*

Una de las razones principales por las cuales se desarrolla este trabajo, es el proveer una comparación justa entre bases de datos orientadas a grafos. La característica principal para satisfacer dicha equidad en la evaluación es instalar y configurar los sistemas en el mismo entorno, y correr los mismos algoritmos con conjuntos de datos idénticos.

Al implementar la metodología que hemos descrito, es importante aclarar que selección se hizo para el entorno de ejecución que satisface las necesidades mínimas de hardware y software para cada base de datos. Dichos requerimientos son obtenidos de los sitios oficiales de cada producto. En el experimento de este trabajo, se utilizaron el mismo tipo de instancias de *Amazon EC2*; cada instancia de hardware es provisionada con la misma cantidad de CPUs, memoria RAM, discos duros, e interfaces de red. El sistema operativo seleccionado debe ser uno en donde se soporte de forma nativa la instalación de casa base de datos orientada a grafos; para nuestras ejecuciones se utilizó *Ubuntu* 18.04 LTS.

Finalmente, al ejecutar este paso, es necesario tener una selección de conjuntos de datos que se utilizarán para las pruebas de carga de datos, y el desempeño en distintos algoritmos para grafos. En esta parte del trabajo, no hay ningún requerimiento que nos obligue a utilizar un conjunto de datos en especial, se recomienda que la selección de datos sea el adecuado para representar el tipo de grafos con el cual los usuarios van a trabajar cuando se implemente la aplicación o trabajo de investigación. Para el caso de estudio de este trabajo se utilizaron los conjuntos de datos descritos en la Tabla 4-2 Conjunto de datos utilizados para aplicar metodología, los cuales también son utilizados por el equipo de *TigerGraph* en su trabajo comparativo de BDGs.

La base de datos obtenida de *Twitter* tiene más de 41 millones de nodos y casi 1.5 billones (miles de millones) de conexiones; este conjunto de datos representa relaciones entre usuarios de la red social. En el caso de los datos de *Graph500* es un grafo creado de forma sintética que tiene 2.4 millones de nodos, y 67 millones de conexiones entre los datos. En ambos casos el grafo en su forma nativa esta formateado como una lista de relaciones entre nodos separadas por un tabulador. Y en ningún conjunto de datos existen propiedades o atributos para los nodos o sus relaciones.

Tabla 4-2 Conjunto de datos utilizados para aplicar metodología

Dataset	Descripción	Vértices	Aristas	Tamaño (MB)
Twitter	Relación entre usuarios de la red social como un grafo dirigido [34]	41.6M	1.47B	24,375
Graph500	Grafo sintético *Kronecker* [35]	2.4M	67M	967

4.3.2 Pruebas de carga de datos

Los objetivos principales de estas pruebas son:

- El tiempo que le toma a la base de datos para cargar el conjunto de datos
- El espacio en disco duro que utilizan para almacenar la información.

En cada sistema de base de datos para grafos a ser analizado, se utiliza el mejor mecanismo que proveen para la carga de datos masivos. En ocasiones, algunos desarrolladores recomiendan hacer cierto pre procesamiento de los datos para poder mejorar los tiempos de carga. En caso de que se haga tal pre procesamiento, se debe mencionar de forma explícita dentro de los resultados oficiales al aplicar la metodología.

Como ya se ha mencionado en parte de la metodología, se recomienda hacer automatización del pre y post procesamiento de datos, de esta manera se puede obtener una forma adecuada de medir el tiempo que estos procesos toman y sumarlos al tiempo que utiliza la base de datos para cargar el conjunto de datos. En el caso de estudio que analizaremos más adelante, dado que todas las bases de datos seleccionadas tienen soporte nativo para el sistema operativos basado en *Linux*, medimos la ejecución de los scripts a través del comando "*time*", el cual arroja estadísticas al correr una instrucción de principio a fin del tiempo utilizado a nivel usuario, sistema y modo *kernel*. Una vez que se siguen los mecanismos que propone el desarrollador o comunidad responsable de la base de datos, se capturan los datos en una tabla para poder ser comparados con las demás bases de datos.

Una vez que se ejecuta la carga de datos, se procede al segundo punto de este proceso, el cual consiste en explorar el tamaño que utiliza la base de datos para almacenar la información. La principal razón por lo cual consideramos importante este paso es porque mucho de los productos e investigaciones que utilizan estos sistemas, suelen tener la necesidad de estar cargando datos constantemente y a escalas grandes, y se tiene que tomar en cuenta el costo que tiene cada sistema al almacenar grandes cantidades de datos. Y así, poder calcular el espacio en disco que cada base de datos requiera a largo plazo. Y aunque se sabe que los *datacenters* actuales tienen soporte para escalar el almacenamiento de datos, el otro problema puede surgir a la hora de acceder a ellos y el espacio en memoria dada la paginación o uso de cache. El cálculo del espacio en disco que utiliza cada sistema BDG puede variar, en ocasiones los desarrolladores proveen de pasos para obtener o calcular este dato de forma muy sencilla, en otras ocasiones se puede complicar y se tiene que recurrir a mecanismos como obtener espacio en disco utilizado antes y después de aplicar la carga de datos. La información es capturada de forma similar al tiempo de carga, en una tabla donde se

pueda comparar con el resto de los sistemas. El siguiente paso de la metodología es obtener el desempeño de las bases de datos al ejecutar consultas sobre los datos.

4.3.3 Pruebas de desempeño de consultas sobre el grafo.

La consulta de datos es la función principal de las bases de datos, y esto no es la excepción para las que están orientadas a grafos. Deben proveer los lenguajes adecuados para realizar cualquier tipo de consultas que un usuario requiera para satisfacer las necesidades de una aplicación. Y por lo mismo es crucial que los resultados obtenidos del sistema sean las adecuadas en tiempo y forma para satisfacer los requerimientos básicos del usuario.

En esta sección de la metodología se hacen pruebas de desempeño en las siguientes áreas:

- Búsqueda de caminos: En esta área, se propone utilizar el tiempo de respuesta al aplicar el algoritmo *k-hop-path*. El cual hace un cálculo del número de nodos vecinos a una cantidad determinada de "saltos. El algoritmo es aplicado a un número aleatorio de nodos.
- Detección de comunidades: Aquí se mide el tiempo de respuesta del algoritmo *Weakly Connected Components* (WCC). Es aplicado a todo el grafo.
- Centralidades: Se mide el tiempo de respuesta del algoritmo *PageRank* (PR). También se aplica al grafo completo.

4.4 Análisis de bases de datos para grafos seleccionadas

La Tabla 4-3 Comparativa de características entre BDGshace un resumen de las características principales de implementación y lanzamiento de cada uno de los sistemas que serán mencionados en las siguientes sub-secciones, de igual manera incluye algunos de los clientes que utilizan el motor de base de daos para solucionar algún servicio o análisis de datos en sus sistemas de producción.

4.4.1 GraphDB

GraphDB es un motor de base de datos diseñado para la web semántica, utiliza tripletas que cumplen con las especificaciones del *Resource Description Framework* (RDF). Está optimizada para el manejo de datos maestros y metadatos. Permita integración con la ingesta de datos en otros formatos como el tabular.

4.4.2 JanusGraph

JanusGraph es una base de datos para grafos distribuida, pertenece al open source bajo *The Linux Foundation*. Utiliza otros proyectos para ciertoas áreas, por ejemplo, en el almacenamiento puede utilizar *Apache Cassandra, HBase* y *Google BigTable*; en el análisis y reporte de datos puede utilizar *Apache Spark, Giraph* o *Hadoop*. La escalabilidad y rendimiento dependen directamente del uso y configuración de dichos sistemas que son utilizados detrás de la base de datos.

4.4.3 Neo4j

Neo4j es una base de datos nativa para grafos NoSQL mantenida en *open source* que provee un *backend* que cumple con las características ACID *(Atomicity, Consistency, Isolation, Durability)* para cualquier aplicación. Es referida como base de datos nativa para grafos porque implementa la propiedad de grafo modelo de forma eficiente hasta el almacenamiento.

4.4.4 TigerGraph

TigerGraph es una plataforma de base de datos para analizar grafos, inspirada en resolver problemas de escalabilidad y procesamiento de datos altamente conectados en tiempo real, a través del uso de programación en paralelo.

Tabla 4-3 Comparativa de características entre BDGs

	GraphDB	JanusGraph	Neo4j	TigerGraph
Fecha de lanzamiento	2000	2017	2007	2017
Lenguaje de implementación	Java	Java	Java	C++
Lenguaje de consultas principal	SPARQL	Gremlin	Cypher	GSQL
Aplicaciones reales	BBC Springer Nature S&P Global	Celum Compose RedHat FiNC	Walmart Adobe Ebay Volvo UBS	China Mobile Intuit VISA Wish Zillow

4.5 Configuración de ambientes de pruebas

El *benchmark* tendrá como ambiente de validación principal un conjunto de instancias *Amazon EC2* tipo *r4.8xlarge*. Dentro de las características de esta clase de instancias se encuentra las siguientes:

- 32 vCPU (*CPU* Virtuales). Unidades de capacidad de cómputo.
- 53 ECU. Medida relativa de potencia de procesamiento íntegra de una instancia de Amazon EC2. Una unidad ECU equivale a la capacidad de CPU de un procesador 2007 *Opteron* o 2007 *Xeon* de 1.0-1.2 GHz
- 244 GiB Memoria RAM. Cantidad de memoria asignada a la instancia de la base de datos.
- 512 GBs de disco duro de estado sólido (SSD), optimizados para operaciones de entrada y salida (IOPS).
- Desempeño de red hasta 10 Gbps (gigabits por segundo).
- Sistema operativo *Ubuntu* 18.04 LTS

4.6 Modelo de datos para grafos

Dentro de las BDG hemos identificado un par de modelo de datos diferentes, el primero está basado en el almacenamiento de tripletas basados en RDF; el segundo es conocido como los *Labeled Property Graphs* (LPG). Ambos proveen formas de describir y explorar conjuntos de datos conectados, pero ambos tienen objetivos distintos, cada uno de estos modelos tienen puntos fuertes dependiendo del caso de uso que se quiera explorar [36].

4.6.1 *Resource Description Framework* (RDF)

RDF es un modelo estándar para el intercambio de datos en la red; facilita la combinación de daos que tienen esquemas diferentes. Entiende la estructura de enlaces en la web al utilizar identificadores de recursos llamados *Uniform Resource Identifiers* (URIs) para nombrar a las relaciones que existen entre elementos y a la conexión que existe entre dichos elementos, a esta nomenclatura se le conoce como

tripleta. La estructura que provee RDF forma un grafo dirigido y con ciertas etiquetas donde las aristas representan una conexión entre dos recursos.

4.6.2 *Labeled Property Graphs* (LPG)

LPG es un modelo el cual enfoca sus capacidades en el almacenamiento y formas para optimizar las consultas y recorridos de grafos que tienen alta conectividad. Los vértices generalmente son referidos como nodos, los cuales tienen un identificador único y un conjunto de propiedades con forma de llave-valor, dichas propiedades definen las características del nodo. De la misma forma, las conexiones o aristas que se forman entre nodos tienen un identificador único y propiedades que definen las características de las relaciones.

En conclusión, los grafos formados con el modelo RDF suelen ser utilizados cuando hay intercambio de información en entornos como lo es internet, y los grafos que se basan en el modelo LPG tienden a ser utilizados cuando el almacenamiento y consultas a los grafos tienen mayor importancia.

4.7 Soporte a lenguajes de consultas

Al saber la importancia que tienen los lenguajes de consulta dentro de una base de datos, se sabe que, sin dichos lenguajes, la base de datos se convierte en un simple sistema de almacenamiento y dejarían de cumplir su función principal. Uno de los principales problemas que tienen el día de hoy las bases de datos para grafos es que prácticamente cada una de ellas provee una implementación de lenguaje diferente. Por lo cual existe muy poca compatibilidad entre motores de bases de datos. Así que una de las principales preguntas que existen actualmente es saber si en algún momento alguno de estos lenguajes pueda jugar el mismo papel que hoy en día tiene SQL para los sistemas de bases de datos relaciones, o si la tendencia será que los distintos motores provean de interfaces que faciliten migrar aplicaciones de un lenguaje a otro. En la Tabla 4-4 Bases de datos vs lenguajes de consulta para grafos, se provee un resumen de los lenguajes que cada BDG utilizada es capaz de utilizar.

Distintas comparaciones han concluido en que la mayoría de los lenguajes para consulta de grafos tienen mejor rendimiento que consultas hechas con SQL para datos masivos y no estructurados [37]. Además de que ya cubren con muchas de las ventajas que solía tener SQL como tener interfaces de programación en lenguajes para sistemas computacionales a la hora de desarrollar software que quiera comunicarse con las BDG.

Las siguientes secciones describen de forma breve algunos de los lenguajes para consultas a grafos más utilizados.

4.7.1 *SPARQL*

Es un lenguaje y protocolo para acceder datos en el modelo RDF. El lenguaje es considerado como orientados a datos ya que solo realiza consultas a la información propia del modelo. El lenguaje no tiene mecanismos para inferir.

4.7.2 *Gremlin*

Es un lenguaje mantenido dentro del proyecto *Apache Tinker-pop* (una plataforma para el computo de grafos). Fue diseñado como un lenguaje para el recorrido de grafos. al ser un lenguaje funcional y orientado al flujo de datos, habilita a usuarios a expresar recorridos y consultas complejas dentro de una aplicación

con el modelo LPG. Cada uno de los pasos que se forman con *Gremlin* son operaciones atómicas en el flujo de datos, y cada una de las operaciones termina formando ya sea un mapa (transformando los objetos en un flujo de datos), un filtro (removiendo objetos de un flujo de datos), o un efecto secundario (cómputo de estadísticas sobre un flujo de datos).

4.7.3 *Cypher*

Es un lenguaje de consultas para grafos el cual permite a los usuarios el almacenar y recuperar datos de una base de datos. Durante un tiempo el lenguaje estuvo limitado a ser usado solo dentro de Neo4j, pero actualmente es un lenguaje que forma parte de la comunidad *open source* bajo el proyecto *openCypher*. Con esta apertura se ha logrado crea una especificación y un kit de compatibilidad técnica. También se tiene acceso a la implementación de los módulos que hacen la interpretación de consultas, la planeación de sentencias y ejecución de estas.

4.7.4 *GSQL*

Es un lenguaje para la exploración y análisis de grafos de escala grande. Provee a los usuarios la capacidad de correr consultas complejas para recurridos de grafos. *GSQL* combina características de otros lenguajes lo cual le da cierta familiaridad a los usuario y programadores de aplicaciones. Su uso está limitado al ser solo soportado por *TigerGraph*.

Un elemento adicional por considerar cuando se exploran las capacidades de cada uno de los lenguajes de consulta es saber si dichos lenguajes son considerados completos bajo la definición de *Turing*, lo cual significa que computacionalmente el lenguaje es equivalente a lo que una máquina de *Turing* puede lograr. O, en otras palabras, quiere decir que el lenguaje es capaz de resolver cualquier problema que una máquina de Turing puede resolver con una cantidad finita de recursos. Los siguientes lenguajes son los que cumplen con las características para poder ser llamados completos bajo la definición de *Turing*: *Cypher*, *Gremlin* y *GSQL*.

La Tabla 4-4 Bases de datos vs lenguajes de consulta para grafos provee una relación de los lenguajes descritos en esta sección y cuales BDG los soportan.

Tabla 4-4 Bases de datos vs lenguajes de consulta para grafos

	Cypher	Gremlin	GSQL	SPARQL
GraphDB	×	×	×	✓
JanusGraph	×	✓	×	×
Neo4j	✓	✓	×	×
TigerGraph	×	×	✓	×

4.8 Interfaces para programación

Muchas aplicaciones de software suelen utilizar bases de datos, en el caso de las especializadas en grafos esto no es diferente. Es decir, se construyen interfaces para los usuarios finales con las cuales ellos pueden interactuar de forma amigable con los datos que administra la base de datos. Dentro de las funciones más comunes se encuentran la capacidad de insertar nueva información, o realizar búsquedas con filtros. Dado que dichos usuarios no tienen el tiempo para aprender a manipular dicha información a través de lenguajes de consultas o la administración de la base de datos, entonces necesitan que dicha aplicación construida y que interactúa con el sistema tenga la capacidad de comunicarse con la base de datos y realizar dichas

operaciones. Dado estas condiciones, se ha hecho un breve análisis de las interfaces para programación o comunicación con la base de datos a través de sus distintos protocolos que son soportados de forma nativa por cada sistema.

Desde una perspectiva de ingeniería de software, la integración con distintos lenguajes es importante a la hora de elegir una base de datos para poder interactuar con los grafos. La Tabla 4-5 Soporte a lenguajes de programación, muestra los lenguajes de programación que soportan cada una de las bases de datos evaluadas:

Tabla 4-5 Soporte a lenguajes de programación

	GraphDB	JanusGraph	Neo4j	TigerGraph
.NET	×	✓	✓	×
Java	✓	✓	✓	✓
JavaScript	×	×	✓	×
Python	×	✓	✓	×
C/C++	×	×	✓	✓
Go	×	✓	✓	×
Ruby	×	✓	✓	×
PHP	×	✓	✓	×
Clojure	×	×	✓	×
R	×	×	✓	×
Spring	×	×	✓	×
Perl	×	×	✓	×
Haskell	×	×	✓	×
Erlang/Elixir	×	✓	✓	×
Scala	×	✓	×	×

5 RESULTADOS Y DISCUSIÓN

Siguiendo la metodología que hemos propuesto en este trabajo, nos va a permitir evaluar las bases de datos para grafos que fueron seleccionadas para el caso de estudio. Y al mismo tiempo nos ayuda a saber si los pasos propuestos son suficientes para concluir con las ventajas y desventajas de cada BDG.

Como ya se ha mencionado, las bases de datos que se evalúan son *GraphDB*, *JanusGraph*, *Neo4j*, y *TigerGraph*. En cada una de las bases de datos se evaluarán aspectos sobre su capacidad de carga de datos con el espacio de almacenamiento que utilizan. Y posteriormente se ejecutan diversos algoritmos para grafos con los cuales podemos medir la efectividad en distintas áreas.

Los conjuntos de datos a utilizar son los descritos en la sección anterior, uno siendo una representación real de interacciones de usuario dentro de *Twitter*, y el otro una base de datos generada por *Graph500* para evaluar a las maquinas que procesan grafos a escala con mejor eficiencia.

Las instancias de hardware y software son configuradas de la misma manera para cada ejecución. Los detalles pueden ser consultados en la sección anterior.

5.1 Resultados

5.1.1 Mecanismos para carga de datos masivos

Dentro de la metodología propuesta, este fue el primer cuestionamiento. Se asume que cada BDG tiene soporte para carga masiva de datos, pero se necesita explorar las formas en que cada uno maneja la ingesta de datos masivos. A continuación, una descripción de la herramienta o interfaz que cada base de datos provee:

- *GraphDB*: Expone múltiples interfaces para la carga de datos en formatos RDF. Dado que los datos que se eligieron para este trabajo y que son comunes a la hora de generar datos son archivos CSV o separados por tabuladores, se ha decidido utilizar el mecanismo *OntoRefine*. Si los datos pueden ser convertidos en una tripleta RDF previo a su carga, entonces se puede utilizar el *endpoint* de SPARQL el cual no tiene límites para el tamaño de carga y utiliza mecanismos de programación en paralelo.
- *JanusGraph*: Provee una interfaz con Java el cual utiliza la API de *Tinkerpop* para agregar nodos y sus relaciones.
- *Neo4j*: El mejor mecanismo para cargar datos masivos es su herramienta neo4j-admin, el cual se considera como una carga "fuera de línea". El desempeño suele ser mejor haciendo un pretrabajo al crear índices sobre los nodos y aristas.
- *TigerGraph*: Carga de datos con un trabajo declarativo que utiliza GSQL con un mecanismo "en línea" usando abultamiento de datos.

Al hacer uso de las interfaces o herramientas mencionadas en el párrafo anterior, se obtuvieron los siguientes datos para cada uno de los sistemas, el tiempo de carga está reflejado en el total de segundos consumidos desde lanzar la instrucción de carga, hasta que el proceso finalizó. Esto se puede observar en la primera fila de la Tabla 5-1 **Tiempo de carga de datos para dataset de *Twitter* y *Graph500***. La tabla también tiene una segunda fila donde se hace la normalización del tiempo de carga utilizando como base 1 al sistema que ejecutó el proceso con la mayor velocidad.

Tabla 5-1 Tiempo de carga de datos para dataset de *Twitter* y *Graph500*

Carga de Datos	**GraphDB**		**JanusGraph**		**Neo4j**		**TigerGraph**	
	twitter	graph500	twitter	graph500	twitter	graph500	twitter	graph500
Segundos	45,240	1,709.2	14,301	1,153.9	4,441	362	2,417	221
Norm.	18.7	7.7	5.9	5.2	1.8	1.6	1.0	1.0

El tiempo de carga de datos para cada uno de los conjuntos de datos con los resultados normalizados están graficados en la Figura 5-1 Resultados de tiempo de carga de datos.

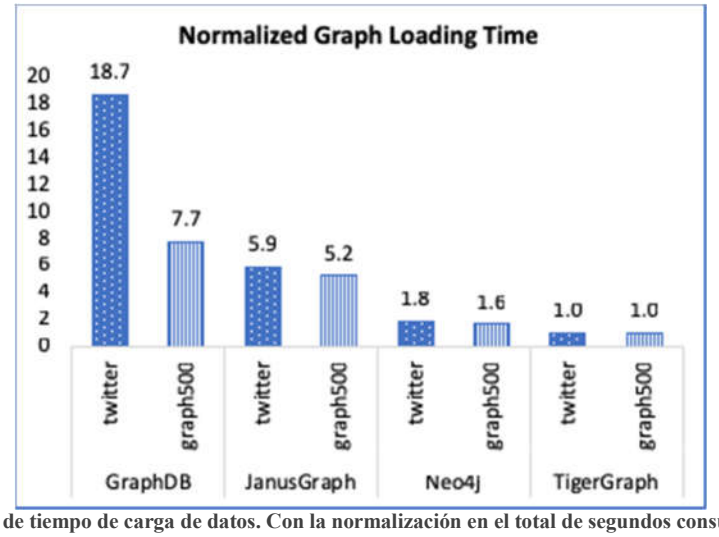

Figura 5-1 Resultados de tiempo de carga de datos. Con la normalización en el total de segundos consumidos desde lanzar la instrucción de carga, hasta que el proceso finalizó. El uno es el de menor tiempo.

La comparativa muestra que *TigerGraph* y *Neo4j* cuentan con un desempeño similar en ambos casos. Debido al formato utilizado, Neo4j requirió un trabajo de indexación previo, el cual no utiliza tiempo de ejecución, pero si requiere un paso adicional en el proceso. En el caso de *TigerGraph*, no hubo necesidad de realizar pasos adicionales. *JanusGraph* muestra que es alrededor de 5 veces más lento que *Neo4j* y *TigerGraph* en ambos conjuntos de datos. Finalmente, *GraphDB* es 7 veces más lento en el caso de datos de *Graph500*, y en el caso de *Twitter* que tiene más datos, toma más de 18 veces de tiempo al cargar los datos.

El otro aspecto que se midió es el espacio en disco que utilizan las BDG para almacenar los datos y sus relaciones, como ya se había mencionado anteriormente en la Tabla 4-2 Conjunto de datos utilizados para aplicar metodología, el tamaño de los datos en su forma original son los siguientes: Graph500 consume 967 MB, un poco menos de un Gigabyte, mientras que Twitter consume 24.3 Gigabytes. En la Tabla 5-2 **Espacio en disco utilizado para dataset de *Twitter* y *Graph500***, podemos observar la primera fila con el espacio en disco total que utiliza cada sistema al cargar cada conjunto, también tiene una segunda fila donde se hace la normalización del espacio en disco utilizado usando la base 1 con la base de datos que utiliza el menor espacio.

Tabla 5-2 Espacio en disco utilizado para dataset de *Twitter* y *Graph500*

Espacio en disco	**GraphDB**		**JanusGraph**		**Neo4j**		**TigerGraph**	
	twitter	graph500	twitter	graph500	twitter	graph500	twitter	graph500
Gigabytes	49	3.85	33	2.5	30	2.3	9.5	0.48
Norm.	5.2	8.0	3.5	5.2	3.2	4.8	1.0	1.0

El único de estos sistemas que utiliza menos espacio en disco que el conjunto de datos en su estado natural es *TigerGraph*, mientras que los demás utilizan más espacio que lo que mide el conjunto de datos en su

estado natural. La gráfica comparativa con los datos normalizados está disponible en la Figura 5-2 Resultados de espacio de almacenamiento utilizado.

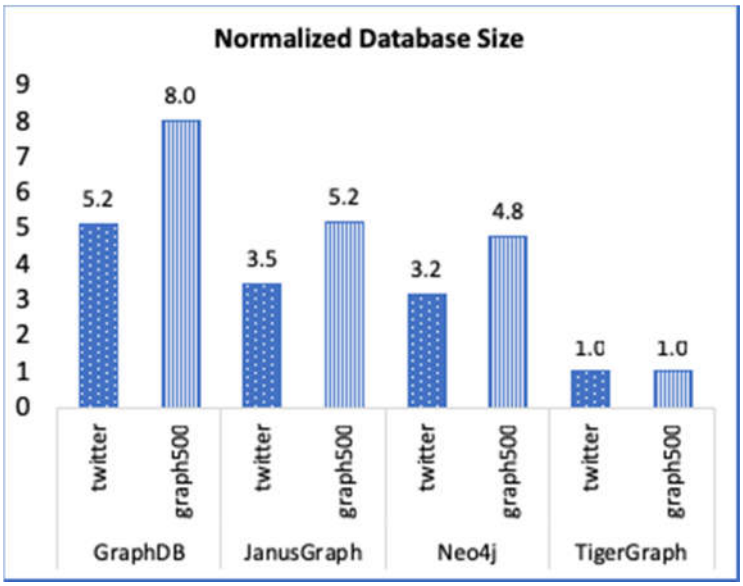

Figura 5-2 Resultados de espacio de almacenamiento utilizado. Con la normalización en el total de megabytes utilizados al concluir la carga de datos. El uno es el de menor tiempo.

5.1.2 Soporte a algoritmos para grafos

Previo al paso en donde se evalúan distintos algoritmos para grafos, la metodología propone hacer una investigación sobre el soporte nativo para su ejecución. Los algoritmos para grafos son utilizados para obtener métricas sobre el grafo, sus nodos y las relaciones entre ellos. Nos pueden proporcionar de información que nos ayuda a conocer que elementos de un grafo son los más relevantes o como su estructura forma diferentes tipos de comunidades.

Como ya se ha discutido, los algoritmos que son propuestos por la metodología tratan de ir más allá de consultas básicas que definen si un nodo está conectado con otro, con la propuesta se logra hacer un recorrido de grafos y detectar características que nos ayuden en aplicaciones o investigaciones complejas. Por lo tanto, este paso de nuestra metodología se hace la pregunta sobre el soporte que tiene de forma nativa para aplicar algoritmos avanzados y optimizados para grafos. Se considera un paso crítico ya que no solo necesitamos consultar relaciones, pero también en ocasiones se necesita hacer cierto tipo de transformaciones para obtener los resultados adecuados.

El lugar para comenzar a revisar dicho soporte comienza con las capacidades que tiene cada lenguaje de consultas que la BDG soporta. Ya se ha mencionado cuáles son los lenguajes de consultas que cada BDG soporta, pero no se ha especificado como operan. Dependiendo de la base de datos, las consultas pueden ser inyectadas al sistema por diversos métodos como scripts, interfaces gráficas, operaciones con servidores *RESTful*, ente otros.

Al iniciar la exploración en las BDG de este caso de estudio encontramos lo siguiente. Para el caso de *GraphDB*, nos dimos cuenta de que es un motor especializado para almacenar grafos en formato RDF; y el único lenguaje que soportan es SPARQL. Este lenguaje tiene un objetivo muy específico el cual se basa en búsqueda entre datos conectados a través de la web semántica. Suele ser muy eficiente cuando se

definen ontologías estructuradas, pero dadas las limitaciones de SPARQL no satisface los requerimientos mínimos que el trabajo propone encontrar dentro de las bases de datos para grafos los cuales nos permitan identificar comunidades, iterar y recorrer el grafo completo, transformar datos, etc. Es por eso por lo que se decide descartar a *GraphDB* y no se aplicarán más pasos dentro del proceso de evaluación.

El caso de *JanusGraph*. Tiene soporte nativo para el lenguaje de consultas *Gremlin*, el cual es parte de una plataforma construida por la comunidad Apache para satisfacer necesidades complejas en la exploración de grafos. Por lo tanto, *JanusGraph* provee de distintas implementaciones de algoritmos para grafos en diversas áreas. Provee mecanismos para calcular centralidades, búsqueda de caminos más cortos, administración de estructuras como árboles, obtener ancestros en común de un conjunto de nodos, recorridos en profundidad y amplitud, entre otros. También tiene la capacidad de detectar ciclos en un grafo, duplicidad de nodos o de aristas. Por lo tanto, *JanusGraph* cuenta con los elementos necesarios para poder continuar con su evaluación.

Neo4j presenta una librería con la implementación de distintos algoritmos para grafos, dicha librería está expuesta como llamadas nativas a procesos de *Cypher*. La lista expuesta de la librería abarca distintas áreas como centralidades, detección de comunidades, búsqueda de caminos, similaridades y predicción de conexiones. A pesar de ser un *plugin* que no está instalado por default en Neo4j, el proceso de habilitarla es muy sencillo y tiene soporte en el sitio oficial de la base de datos. Se puede continuar con la evaluación de Neo4j al contar con los requerimientos mínimos.

Similar a lo descrito por *Neo4j*, el equipo de *TigerGraph* provee de una librería de algoritmos programados con GSQL que puede ser utilizado dentro de la base de datos; tiene un soporte que incluye áreas cono centralidades, detección de comunidades, búsqueda de caminos, entre otros. Por lo tanto, *TigerGraph* es una base de datos con la cual se puede continuar en el proceso de evaluación.

La Tabla 5-3 **Soporte nativo de algoritmos para grafos en cada base de datos seleccionada** muestra un resumen del soporte de distintas áreas de los algoritmos para grafos que son soportados de forma nativa por cada una de las bases de datos seleccionadas.

Tabla 5-3 Soporte nativo de algoritmos para grafos en cada base de datos seleccionada

	GraphDB	JanusGraph	Neo4j	TigerGraph
Centralities	×	✓	✓	✓
Community Detection	×	×	✓	✓
Path Finding	×	✓	✓	✓
Similarity	×	×	✓	✓
Link Prediction	×	×	✓	×
Others	×	×	✓	×

5.1.3 Desempeño en consultas sobre grafos

Ya que sabemos con cuales de las BDG se puede continuar, se procede al paso de pruebas de los algoritmos en cada una de ellas. Como ya fue descrito en la metodología, las pruebas miden el desempeño con estos criterios:

- Tiempo de respuesta en consultas para el algoritmo de *k-hop-path*.
- Tiempo de respuesta al aplicar el algoritmo *Weakly Connected Components*

- Tiempo de respuesta al aplicar el algoritmo *PageRank*.

Cada una de estas consultas son implementadas en el lenguaje que soportan de forma nativa; *Gremlin* para *JanusGraph*, *Cypher* para *Neo4j*, y GSQL para *TigerGraph*.

En las siguientes subsecciones se presentan tablas de resultadas donde por lo general se toma como referencia el tiempo de ejecución de cada algoritmo para los distintos conjuntos de datos aplicados a cada BDG. Dentro de esas tablas se utiliza el término normalización, este dato se calcula con una simple regla de tres, donde se utiliza el mejor de los tiempos (menos segundos consumidos), y se le asigna el valor de uno (1); después para el resto de los valores se aplica la multiplicación por el valor base (uno como definimos anteriormente) y de divide entre el tiempo al que corresponde el uno. Como ejemplo si el valor mínimo en un resultado entre BDG es de 20 segundos, le asignamos el valor 1, si el siguiente valor es de 40 segundos se hace la operación (40 * 1) / 20 y nos regresa un valor de 2. Con este dato podemos deducir que el primer resultado es 2 veces (2X) mejor que el segundo.

5.1.3.1 K-Hop-Paths (K-Neighborhood)

El algoritmo hace una cuenta y pregunta por el total de vértices que tiene una longitud k desde un vértice inicial. Ayuda a medir el desempeño en recorridos de grafos.

La configuración utilizada para k igual a uno y dos (*1-hop* y 2-hop):

- Nodos preseleccionados: 300
- Tiempo fuera: 3 minutos por consulta
- Tamaño del vecindario para k=1 en graph500: 5.13E+03
- Tamaño del vecindario para k=1 en twitter: 1.06E+05
- Tamaño del vecindario para k=2 en graph500: 4.89E+05
- Tamaño del vecindario para k=2 en twitter: 3.25E+06

Los resultados para cada una de las BDG con los saltos igual a uno y dos, están disponibles en las tablas Tabla 5-4 **Resultados *K-Neighborhood* con k igual a 1** y Tabla 5-5 Resultados *K-Neighborhood* con k igual a 2.

Tabla 5-4 Resultados *K-Neighborhood* con k igual a 1

One-Hop	Medición	JanusGraph	Neo4j	TigerGraph
Graph500	tiempo (seg)	0.0195	0.018	0.003
	normalización	7	6	1
	tiempo fuera (%)	0%	0%	0%
Twitter	tiempo (seg)	1.42	0.377	0.017
	normalización	84	22	1
	tiempo fuera (%)	0%	0%	0%

Tabla 5-5 Resultados *K-Neighborhood* con k igual a 2

Two-Hop	Medición	JanusGraph	Neo4j	TigerGraph
Graph500	tiempo (seg)	13.95	4.4	0.066
	normalización	211	67	1

	tiempo fuera (%)	0%	0%	0%
Twitter	tiempo (seg)	40.398	5.26	0.33
	normalización	122	16	1
	tiempo fuera (%)	0%	0%	0%

La configuración utilizada para k igual a tres y seis (*3-hop* y *6-hop*):

- Nodos preseleccionados: 10
- Tiempo fuera: 2.5 horas por consulta
- Tamaño del vecindario para k=3 en graph500: 1.43E+06
- Tamaño del vecindario para k=3 en twitter: 2.07E+07
- Tamaño del vecindario para k=6 en graph500: 1.58E+06
- Tamaño del vecindario para k=6 en twitter: 3.50E+07

Los resultados para cada una de las BDG con los saltos igual a tres y seis, están disponibles en las tablas Tabla 5-6 Resultados *K-Neighborhood* con k igual a 3 y Tabla 5-7 Resultados *K-Neighborhood* con k igual a 6.

Tabla 5-6 Resultados *K-Neighborhood* con k igual a 3

Three-Hop	Medición	JanusGraph	Neo4j	TigerGraph
Graph500	tiempo (seg)	1965.1	58.5	0.405
	normalización	4852	144	1
	tiempo fuera (%)	10%	0%	0%
Twitter	tiempo (seg)	1600.6	99.7	2.9
	normalización	552	34	1
	tiempo fuera (%)	10%	30%	0%

Tabla 5-7 Resultados *K-Neighborhood* con k igual a 6

Six-Hop	Medición	JanusGraph	Neo4j	TigerGraph
Graph500	tiempo (seg)	N/A	1583.1	1.79
	normalización	-	884	1
	tiempo fuera (%)	100%	80%	0
Twitter	tiempo (seg)	N/A	N/A	20.6
	normalización	-	-	1
	tiempo fuera (%)	100%	100%	0%

TigerGraph muestra el mejor desempeño en cualquier tamaño de k-hop, y fue la única base de datos capaz de realizar consultas cuando k se le asigna un valor de seis sin haber fallado por tiempos fueras u otro tipo de problema con la plataforma de pruebas. En los casos donde la tabla muestra entradas como N/A (*Not Available*) nos referimos a que la base de datos no fue capaz de terminar el procesamiento del algoritmo en un tiempo menor a la configuración del *benchmark*, dado a encontrar este tipo de entradas en k. igual

a seis, el tiempo fuera fue configurado con 2.5 horas por cada consulta para los diez nodos seleccionados de forma aleatoria.

Cuando k=1 *TigerGraph* es al menos 6 veces más rápido que su competidor más cercano. En k=2 el tiempo de *TigerGraph* es al menos 16 veces más rápido que los demás. En ambos casos *Neo4j* es la BDG que se funge como el competidor más cercano. Y *JanusGraph* es comparable solo cuando k=1 en el conjunto de datos menos grande, después su desempeño es considerablemente menor.

Similar a los resultados pasados, *TigerGraph* es el que mejores resultados obtiene cuando k=3 y k=6. Y es en estos escenarios donde encontramos los primeros casos donde el algoritmo no pude completarse en los tiempos establecidos para algunos casos. Cuando los saltos son configurados como k=3, *TigerGraph* es 34 veces mejor que *Neo4j*. La comparativa de k=6 es incompleta ya que solo *TigerGraph* pudo completar todas las búsquedas. *Neo4j* solo pudo completar 20% de las consultas en *Graph500*. El resto de los intentos fueron incompletos al no terminar en los tiempos definidos.

5.1.3.2 Weakly Connected Components

En este algoritmo se hace un recorrido sobre todo el grafo y realiza el cálculo de resultados para describir características del grafo. El WCC encuentra y etiqueta a todos los componentes del grafo. Requiere de un recorrido completo de vértices y aristas.

Al ejecutar el algoritmo con ambos conjuntos de datos para cada uno de los sistemas, se obtuvieron los números reflejados en la Tabla 5-8 Ejecución del algortimo *Weakly Connected Components*. La primera fila representa los segundos que cada sistema utilizó para ejecutar el algoritmo, en caso de que después de 24 horas la ejecución no haya sido concluida se canceló la ejecución y se marca la tabla con la leyenda N/A (*Not Available*). En la segunda fila se captura el tiempo normalizado otorgando como base 1 al sistema que tomó el menor tiempo en concluir a ejecución.

Tabla 5-8 Ejecución del algortimo *Weakly Connected Components*

Weakly Connected Components	**JanusGraph**		**Neo4j**		**TigerGraph**	
	twitter	graph500	twitter	graph500	twitter	graph500
Tiempo (seg)	N/A	1491.4	1545.1	65.5	47.9	2.88
Normalización	-	518	32	23	1	1

Los resultados con la información normalizada se encuentran graficados en la Figura 5-3 Resultados de ejecución de WCC.

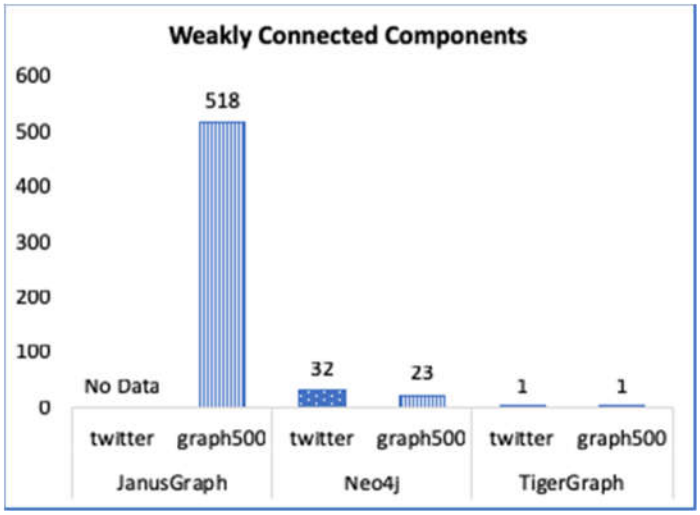

Figura 5-3 Resultados de ejecución de WCC. Con la normalización en el total de segundos consumidos desde iniciar el algoritmo, hasta que el proceso finalizó. El uno es el de menor tiempo.

TigerGraph es 23 veces más rápido que Neo4j y más de 500 veces mejor que *JanusGraph*. "No Data" refiere a que el algoritmo no fue capaz de completarse en 24 horas.

5.1.3.3 PageRank

El algoritmo *PageRank* es un proceso iterativo el cual recorre cada vértice, durante cada iteración hace el cálculo de una calificación para dicho vértice.

Al ejecutar el algoritmo con ambos conjuntos de datos para cada uno de los sistemas, se obtuvieron los números reflejados en la Tabla 5-9 Ejecución del algortimo *PageRank*. La primera fila representa los segundos que cada sistema utilizó para ejecutar el algoritmo, en caso de que después de 24 horas la ejecución no haya sido concluida se canceló la ejecución y se marca la tabla con la leyenda N/A (*Not Available*). En la segunda fila se captura el tiempo normalizado otorgando como base 1 al sistema que tomó el menor tiempo en concluir a ejecución.

Tabla 5-9 Ejecución del algortimo *PageRank*

PageRank	**JanusGraph**		**Neo4j**		**TigerGraph**	
	twitter	graph500	twitter	graph500	twitter	graph500
Tiempo (seg)	N/A	2279.0515	614.9	31.2	166.03	12.8
Normalización	-	178	4	2	1	1

El tiempo de respuesta de este algoritmo con un número de 10 iteraciones y los datos normalizados es observado en la Figura 5-4 Resultados de ejecución de *PageRank*.

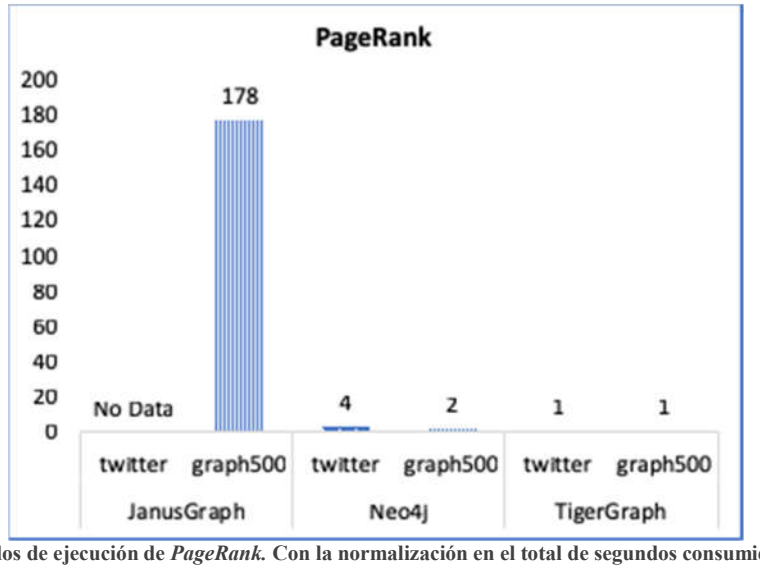

Figura 5-4 Resultados de ejecución de *PageRank.* Con la normalización en el total de segundos consumidos desde iniciar el algoritmo, hasta que el proceso finalizó. El uno es el de menor tiempo

TigerGraph es 2 veces más rápido que *Neo4j* y hasta 178 veces mejor que *JanusGraph*. Con el conjunto de datos de twitter (de mayor tamaño), el desempeño incrementa a 4 veces más eficiente *TigerGraph* que *Neo4j*. "No data" refiere a que el algoritmo no fue capaz de completarse en 24 horas.

5.2 Discusión

Dentro de los resultados obtenidos en este trabajo se puede concluir que las mejoras continuas en las plataformas existentes son claves para poder sobrevivir a las exigencias de la comunidad. Ya que en ocasiones nos encontramos con bases de datos orientadas a grafos que hace una década tomaban una relevancia importante por su uso, pero con el tiempo fueron desplazadas y llevadas a no tener soporte o rebasadas en desempeño por cualquier base de datos nueva que intentara tomar un pedazo del mercado que ya lo estaba utilizando.

En el caso de estudio descubrimos que una de las razones por las cuales *Neo4j* es un sistema que soporta más lenguajes, mayor cantidad de algoritmos y tutoriales es debido a que es el sistema que tiene más tiempo en la comunidad y ha evolucionado en sus versiones al recibir retroalimentación de sus usuarios. Sin embargo, *TigerGraph* y *JanusGraph* son ejemplos de aplicaciones recientes las cuales han aprendido dichas lecciones de la comunidad que utiliza grafos para mejorar desde sus primeras versiones en el soporte que proveen para la carga de datos masivos y el soporte a exploración de grafos. Incluso atacan algunas de las limitaciones actuales de *Neo4j*. El caso de *GraphDB* es uno especial; están posicionados como uno de los mejore sistemas para el caso de uso de tripletas y cumplen de forma estricta con el estándar establecido por el W3C con respecto a RDF y provee de varias herramientas alrededor de dicho caso de uso.

Cuando tuvimos nuestro acercamiento inicial con cada base de datos para llevarla a un estado operacional y aplicar el caso de estudio, fue fácil identificar las dependencias que cada una de las BDG tiene. Para las soluciones que *GraphDB* y *Neo4j* tienen, sus requerimientos están cubiertos por soluciones internas de los desarrolladores, por lo cual hace parecer que todo trabajo como un sistema único. Mientras que en los casos de *JanusGraph* y *TigerGraph* es muy claro que usan soluciones *open source* para ciertas funcionalidades que trabajan por detrás de la base de daos, algunas de estas soluciones de las cuales toman

ventaja son *Cassandra*, *Kafka*, y *Zookeeper*, todas estas son mantenidas por las comunidades *open source*, y están empaquetadas con el instalador de la base de datos con muy poca necesidad de tener que ser configuradas por el usuario.

Sobre el caso del lenguaje de consultas: Es muy probable que ninguno de los lenguajes para consultas sobre grafos pueda consolidarse pronto como el lenguaje estándar para dicha área. La razón principal es que al ser un mundo donde los datos no están estructurados y una gran variedad de formatos, se dificulta escribir un lenguaje que sea capaz de consumir todos. Los usuarios deben conocer las capacidades y limitaciones que tiene cada lenguaje, al elegir uno se debe considerar como transformar el origen de los datos a un formato que le convenga al lenguaje y después desarrollar los casos de uso con el soporte a algoritmos que se tiene.

Analizar la facilidad de uso de las BDG puede convertirse en una tarea que depende de la experiencia del usuario con bases de datos tradicionales. En ninguno de los casos analizados fue complicado llevarlo a un estado operacional óptimo.

Es importante señalar nuevamente que todos los resultados obtenidos y expuestos en este trabajo se realizaron con una sola plataforma de ejecución y utilizando la versión *open source* o gratuita que proveen los desarrolladores. Pueden existir mejoras importantes en cada caso al utilizar plataformas que permiten jerarquías de clúster y en ediciones que requieran de un pago económico.

6 CONCLUSIONES

6.1 Conclusiones

Al comenzar a elegir las bases de datos para grafos más populares en el mercado nos dimos cuenta de que es muy común ver que el desarrollo de este tipo de productos es constante y reciente. Nuevos sistemas han aparecido en los últimos dos años y tratan de obtener atención de la comunidad que utiliza dicha tecnología, su propuesta generalmente viene acompañada de publicaciones en donde arrojan resultados favorecedores sobre las bases de datos más populares, peor no abarcan todo el espectro que dichos sistemas pueden alcanzar. La metodología que proponemos ayuda a que los usuarios interesados en conocer los alcances de las bases de datos para grafos puedan tomar la decisión adecuada para resolver los problemas que sus necesidades requieren. Se logra al exponer, para cada BDG, las características principales, los lenguajes de consulta e interfaces que proveen, el soporte de algoritmos que tienen y el desempeño que muestran al ser configurados con las mismas condiciones. Así se puede identificar sus fortalezas y puntos de mejora para poder elegir el motor de base de datos que satisface de mejor forma los requerimientos que tienen al implementar sus soluciones. La metodología es flexible al permitir distintos conjuntos de datos y algoritmos a aplicar, el caso de estudio seleccionado muestra un ejemplo de ello y puede ser modificado para investigar otras áreas.

Los grafos van a continuar siendo una fuente de datos importante dentro del desarrollo de aplicaciones de software. Ya que han resuelto problemas de distintas ramas de la ciencia al ayudar a entender el comportamiento de datos altamente conectados. Industrias como la financiera, la mercadotecnia, comportamientos sociales, y descubrimientos en áreas médicas son solo algunos de los ejemplos que se encuentran hoy en día donde el procesamiento de grafos ya juega un papel importante para obtener información que revoluciona a las industrias.

La diversidad de bases de datos para grafos aunado al hecho de no tener un lenguaje de consultas estándar puede complicar la llegada de más sistemas ya que no habrá mucho interés en tener gente que constantemente tenga que aprender los lenguajes que cada uno introduce. Por lo cual, al parecer durante los siguientes años se pudiera comenzar a ver una tendencia a ver desaparecer muchos de estos sistemas, y aquellos que logren posicionar su lenguaje de consultas como uno de alto desempeño y sencillo de aprender, pueden posicionarse como los más aceptados en el mercado e incluso convertirse en el estándar. Por el momento concluimos que será difícil que se dé ese caso en un corto o mediano plazo.

6.2 Trabajo Futuro

Como fue descrito a través del trabajo, la metodología desarrollada no está limitada a poder replicar cosos con diferentes entornos de ejecución, y adaptar a las BDG para ser configuradas para obtener los mejores resultados posibles para un conjunto de datos; se requiere de trabajo adicional para ir a entender todos los comportamientos que se pueden obtener a través de la documentación completa que proveen los desarrolladores de casa base de datos, de esta forma se podrían obtener mejores resultados de desempeño y explotar las funcionalidades que cada una tiene.

El trabajo realizado con los conjuntos de datos y algoritmos seleccionados hacen uso de una característica importante del modelo LPG, es decir, trabajar con las propiedades/atributos que pueden asociarse a los vértices y aristas. Se considera que para futuras implementaciones se puede considerar dicho aspecto y elegir casos de uso que ejerzan consultas sobre dichas propiedades de un grafo.

La selección de nuestro caso de estudio abarca a un conjunto de bases de datos populares, pero la realidad es que hay varias BDG que no fueron utilizadas y que pudieran ser exploradas de la misma forma en que se aplica la metodología en este trabajo, por mencionar algunas tenemos a *Dgraph*, *InifinteFraph*, *InfoGrid*, *FlockDB*, y *HyperGraphDB*.

Otra área por explorar son las bases de datos que no son diseñadas para el procesamiento de grafos, pero ya cuentan con mecanismo que soportan ciertas operaciones sobre datos conectados. No existen razones por las cuales dichas bases de datos pudieran tener implementaciones que mejoren los resultados obtenidos sobre las BDG. Algunos ejemplos de este tipo de motores de bases de datos son *ArangoDB*, *OrientDB*, y *Stardog*.

Relacionado al tema del soporte de aplicaciones construidas sobre las BDG, existe una tendencia en la industria a correr dichas aplicaciones en entornos *cloud*. Uno de los proveedores más importantes para servicios web es Amazon a través de su plataforma AWS, y recientemente anunciaron un producto llamado *Amazon Neptune* el cual toma ventajas de plataformas para grafos como *GraphDB* para trabajar con tripletas RDF. Provee de interfaces de fácil uso para realizar consultas sobre datos cargados a través de su servicio, y tienen la ventaja de que se pueden conectar con el resto de los servicios que proveen.

Es importante señalar que el trabajo excluyó a otras plataformas especializadas para el procesamiento de grafos como *Giraph* y *GraphX*. Una de las razones principales por la cual fueron excluidas es porque su uso real es para trabajar con datos en memoria y procesarlos “en línea”. Lo interesante es que algunas de las BDG actualmente proveen de integración con este tipo de plataformas, por lo tanto, a la hora de procesar información con estas integraciones se podrían obtener diferentes resultados en el desempeño de consultas.

La metodología propuesta también puede ser extendida para proveer caminos donde puedan compararse de forma completa aquellas que soportan el modelo RDF. Uno de los cambios sería limitar el consumo de datos como ontologías o crear un paso donde cualquier conjunto de datos pueda ser convertido a tripletas.

BIBLIOGRAFÍA

[1] M. Junghanns, A. Petermann, M. Neumann y E. Rahm, «Management and Analysis of Big Graph Data: Current Systems and Open Challenges,» de *Handbook of Big Data Technologies*, Sydney, Springer, 2017.

[2] B. Elser y A. Montresor, «An Evaluation Study of BigData Frameworks for Graph Processing,» Università degli Studi di Trento, Trento, 2013.

[3] Y. Lu, J. Cheng y H. Wu, «Large-Scale Distributed Graph Computing Systems: An Experimental Evaluation,» Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, 2014.

[4] M. Chen, S. Mao y Y. Liu, «Big Data: A Survey,» de *Mobile Networks and Applications*, New York, Springer US, 2014.

[5] J. Han, H. E y G. Le, «Survey on NoSQL Database,» Beijing University of Posts and Telecommunications, Beijing.

[6] J. Hirsch, «An index to quantify an individual's scientific research output.,» Proc Natl Acad Sci , La Jolla, 2005.

[7] D. J. Watts, «Six Degrees: The Science of a Connected Age,» W. W. Norton & Company, New York, 2004.

[8] D. Kempe, J. Kleinberg y E. Tarods, «Maximizing the spread of influence through a social network,» de *Proceeding KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington DC, ACM New York, 2003, pp. 137-146.

[9] F. Bajaber, R. Elshawi, O. Batarfi, A. Altalhi, A. Barnawi y S. Sakr, «Big Data 2.0 Processing Systems: Taxonomy and Open Challenges,» de *Journal of Grid Computing*, Netherlands, Springer Science+Business Media, 2016, p. 379–405.

[10] L. G. Valiant, «A bridging model for parallel computation,» *Communications of the ACM,* vol. 33, nº 8, pp. 103-111, 1990.

[11] Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella y T. L. Willke, «How Well do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis,» Delft University of Technology, Delft, 2014.

[12] M. Han, K. Daudjee, K. Ammar, M. T. Ozsu, X. Wang y T. Jin, «An experimental comparison of pregel-like graph processing systems,» *Proceedings of the VLDB Endowment,* vol. 7, nº 12, pp. 1047-1058, 2014.

[13] Y. Zhao, K. Yoshigoe, M. Xie, S. Zhou, R. Seker y J. Bian, «Evaluation and Analysis of Distributed Graph-Parallel Processing Frameworks,» *Journal of Cyber Security,* vol. 3, p. 289–316, 2014.

[14] R. McColl, D. Ediger, J. Poovey, D. Campbell y D. A. Bader, «A Performance Evaluation of Open Source Graph Databases,» Georgia Institute of Technology, Atlanta, 2014.

[15] S. Nadathur, N. Sundaram, M. Ali Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin y P. Dubey, «Navigating the maze of graph analytics frameworks using massive graph datasets,» de *roceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Utah, ACM New York, 2014, pp. 979-990.

[16] S. Batra y T. Charu, «Comparative Analysis of Relational And Graph Databases,» *International Journal of Soft Computing and Engineering,* vol. 2, nº 2, 2012.

[17] A. Nayak, A. Poriya y D. Poojary, «Type of NOSQL Databases and its Comparison with Relational Databases,» *International Journal of Applied Information Systems,* vol. 5, nº 4, 2013.

[18] D. Fernandes y J. Bernardino, «Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB,» de *Proceedings of the 7th International COmference on Data Science, Technology and Applications*, 2018.

[19] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vanó, S. Gómez-Villamor, N. Martínez-Bazan y J.-L. Larriba-Pey, «Survey of graph database performance on the hpc scalable graph analysis benchmark,» de *Internation Conference on the Web-Age Information Management*, Berlin, 2010.

[20] Gartner, «Gartner, Inc.,» 5 Octubre 2017. [En línea]. Available: https://www.gartner.com/technology/research/methodologies/magicQuadrants.jsp. [Último acceso: 5 Noviembre 2017].

[21] A. Auld, «Can we really trust the Gartner Magic Quadrant?,» 3 March 2017. [En línea]. Available: https://www.cbronline.com/news/enterprise-it/can-really-trust-gartner-magic-quadrant/. [Último acceso: 5 November 2017].

[22] A. Petermann, M. Junghanns, E. Rahm, K. Gómez y S. Kemper, «Research Gate,» Research Gate, 2 Marzo 2017. [En línea]. Available: https://www.researchgate.net/project/Gradoop. [Último acceso: 11 Noviember 2017].

[23] R. C. Murphy, K. B. Wheeler, B. W. Barret y J. A. Ang, «Introducing the graph 500,» *Cray Users Group,* vol. 19, pp. 45-74, 2010.

[24] . B. Merkl Sasaki, «NEO4J,» NEO4J, 18 Septiembre 2015. [En línea]. Available: https://neo4j.com/blog/other-graph-database-technologies/. [Último acceso: 12 Noviembre 2017].

[25] J. Vaidehi, «DEV,» 21 Marzo 2017. [En línea]. Available: https://dev.to/vaidehijoshi/a-gentle-introduction-to-graph-theory. [Último acceso: 5 Noviembre 2017].

[26] V. Daboin, L. Vázquez, S. Sifuentes y A. Pérez, «emaze,» República Bolivariana de Venezuela Ministerio del Poder Popular para la Educación Carvajal, [En línea]. Available: https://www.emaze.com/@ALZFTQCW/-grafos-copy1. [Último acceso: 2017 Noviembre 2017].

[27] J. Leskovec y A. Krevl, «Stanford Large Network Dataset Collection,» 1 Junio 2014. [En línea]. Available: https://snap.stanford.edu/data/. [Último acceso: 5 Noviembre 2017].

[28] Universitat Koblenz, «Konect Networks,» 27 Abril 2017. [En línea]. Available: http://konect.uni-koblenz.de/networks/. [Último acceso: 5 Noviembre 2017].

[29] University of Colorado, «ICON,» [En línea]. [Último acceso: 5 Noviembre 2017].

[30] Docker, «What is a Container,» [En línea]. Available: https://www.docker.com/what-container. [Último acceso: 12 Noviembre 2017].

[31] B. Adler y K. Weins, «Cloud Migration and Portability,» 23 Agosto 20017. [En línea]. Available: https://www.slideshare.net/rightscale/cloud-migration-and-portability-with-and-without-containers. [Último acceso: 12 Noviembre 2017].

[32] «Method of calculating the scores of the DB-Engines Ranking,» DB Engines, 2019. [En línea]. Available: https://db-engines.com/en/ranking_definition. [Último acceso: 05 05 2019].

[33] «TigerGraph Ecosystem,» TigerGraph, 28 02 2019. [En línea]. Available: https://github.com/tigergraph/ecosys/tree/benchmark. [Último acceso: 05 05 2019].

[34] H. Kwak, C. Lee y S. Moon, «What is Twitter, a Social Network or a News Media?,» de *Proceedings of the 19th International World Wide Web (WWW) Conference*, Raleigh, NC, 2010.

[35] «Sample High-Level Implementation of the Kronecker Generator,» Graph500, 20 June 2017. [En línea]. Available: https://graph500.org/?page_id=12#sec-3_3. [Último acceso: 05 05 2019].

[36] J. Barrasa, «RDF Triple Stores vs. Labeled Property Graphs: What's the Difference?,» Neo4j, 18 08 2017. [En línea]. Available: https://neo4j.com/blog/rdf-triple-store-vs-labeled-property-graph-difference/. [Último acceso: 05 05 2019].

[37] F. Holzschuher y R. Peinl, «Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j,» de *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, ACM, 2013, pp. 195-204.

[38] A. Peterman, M. Junghanns, E. Rahm, K. Gomez y S. Kemper, «Research Gate,» 26 June 2017. [En línea]. Available: https://www.researchgate.net/project/Gradoop.

APÉNDICE A. Tablas comparativas de trabajos relacionados

A continuación, se muestra un resumen de los trabajos relacionados que fueron analizados dentro de la investigación previa al desarrollo de este trabajo. Cada tabla muestra el título del trabajo o estudio junto con los siguientes campos:

- Bases de Datos Evaluadas: Nombre de las bases de datos comparadas.
- Sets de Datos: Conjuntos de datos que fueron utilizados para correr los análisis y/o pruebas.
- Algoritmos: Nombre de los algoritmos aplicados para ejecutar las pruebas y comparaciones.
- Ambiente de validación: Descripción del hardware y software que cada trabajo utilizó para cargar los datos y correr los algoritmos.
- Métricas: Elementos que fueron medidos al ejecutar las pruebas con los datos que se describen en los campos anteriores.

An evaluation study of big data frameworks for graph processing				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Map-Reduce - Stratosphere - Hama - Giraph - GraphLab	- ca.AstroPh - ca.CondMat - Amazon0601 - web-BerkStan - com.Youtube - wiki-Talk - com.Orkut	- K-Core	- Sistema: 32 nodos EC2 M1 Medium con 2 CPUs - RAM: 3.75 GB c/u. ---------- - Sistema: 16 CPU machine - RAM: 10 GB - Almacenamiento: Sata-SSDs.	Tiempo de ejecución.

How well do graph-processing platforms perform? An empirical performance evaluation and analysis				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Hadoop - YARN - Stratosphere - Giraph - Neo4j	- Amazon - WikiTalk - KGS - Citation - DotaLeague - Synth - Friendster	- BFS - STATS - CONN - CD - EVO	- Sistema: DAS4 SC Intel Xeon E5620 2.4 GHz CPU (dual quad-core, 12 MB cache) - RAM: 24 GB - Red: 1 Gbit/s Ethernet - Sistema Archivos: NFS - Sistema Operativo: CentOS 6.3 (kernel 2.6.32)	- Poder de procesamiento - Utilización de recursos del sistema - Escalabilidad - *Overhead*
An experimental comparison of pregel-like graph processing systems.				
Bases de Datos	**Sets de Datos**	**Algoritmos**	**Ambiente de**	**Métricas**

Evaluadas			**validación**	
- Giraph - GPS - Mizan - GraphLab	- soc-LiveJournal1 - com-Orkut - arabic-2005 - twitter-2010 - uk-2007-05	- RW - ST - PT - GM	- 16-128 nodos *Amazon* EC2 m1.xlarge en región uswest-2c. Con 4 CPUs, (1.7 GHz Xeon) - RAM:15 GB - Sistema Operativo: Ubuntu 12.04.1 (kernel 3.2.0-36-virtual).	- Cómputo - Configuración - Tiempo de ejecución - Memoria - Tráfico de red

Evaluation and analysis of distributed graph-parallel processing frameworks.				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- PowerGraph - Giraph - GPS	- BFS1 - Soc-LiveJournal - Com-DBLP	- PR - SSSP - *Triangle Count*	- Sistema: 48 nodos - Resource Mgr: TORQUE - Scheduler: Moab - RAM: 16 GB - CPU: 2 quad-core Intel Xeon 2.66GHz CPUs. - Sistema de archivos: NFS - Sistema Operativo: Linux	- Tiempo de procesamiento de datos - Escalabilidad - Uso de recursos del sistema

A performance evaluation of open source graph databases				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
SQL databases: - SQLite - MySQL - Oracle - Microsoft SQL Server NoSQL: - Neo4j - OrientDB - InfoGrid - Titan - FlockDB - ArangoDB - InfiniteGraph - AllegroGraph	12 conjuntos de datos de licencia *open source* clasificados de pequeños a diminutos, por su cantidad de vértices y aristas	- SSSP - SV - PR	- Sistema: Un solo nodo. - RAM: 256 GB - CPU: AMD Opteron 6282 SE ---------- Pruebas distribuidas - CPU: 2 Intel Xeon X5660 - RAM: 24 GB - Red: QDR Infiniband.	Tiempo de ejecución.

- DEX - GraphBase - HyperGraphDB BSP engines: - Bagel - Hama - Giraph - PEGASUS - Faunus In-memory graph packages: - NetworkX - Gephi - MTGL - Boost - uRiKA - STINGER				

Navigating the maze of graph analytics frameworks using massive graph datasets				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Native - GraphLab - CombBLAS - SociaLite - Galios - Giraph	- Facebook - Wikipedia - LiveJournal - Netflix - Twitter - Yahoo Music - Synthetic Graph500 - Synthetic Collaborative Filtering	- *Page Rank* - *Breadth First Search* - *Collaborative Filtering* - *Triangle Count*	- Sistema: CPU: Intel Xeon 2 CPU E5-2697 - RAM: 64 GB - Red: Mellanox Infiniband - Sistema Operativo: Red Hat Enterprise Linux Server OS release 6.4.	- Uso de CPU - Tamaño de memoria - Características del uso de red

Large-scale distributed graph computing systems: an experimental evaluation				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Pregel - Giraph - GraphLab - GraphX - Mizan	- WebUk - Friendster - Twitter - LiveJournal - BTC	- PR - *Diameter Estimation* -SSSP - Hashmin	- Sistema: 15 maquinas CPU: Intel(R) Xeon(R) E5-2620 CPU	Tiempo de ejecución

- GPS - Giraph++ - Pregelix - Pregel+ - Blogel	- USA Road	- SV - BMM - GC	- RAM: 48 GB - Almacenamiento: SATA (6Gb/s, 10k rpm, 64MB cache) - Red: Broadcom Gigabit Ethernet - Sistema Operativo: 64-bit CentOS 6.5 (kernel 2.6.32)	

Comparative Analysis of Relational and Graph Databases.				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- MySQL - Neo4j	- Bases de amistades y gustos por películas.	- Búsqueda	- Sin especificar	- Soporte y Maduración - Seguridad - Flexibilidad - Tiempo de respuesta en búsquedas relacionales

Type of NoSQL Databases and its comparison with Relational Databases.				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Amazon DynamoDB - RIAK - Big Table - Cassandra - MongoDB - Neo4j - db4o	- Ninguno	- Ninguno	-	- Modelo de datos - Casos de uso

Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4j and OrientDB.				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- AllegroGraph - ArangoDB - InfiniteGraph - Neo4j - OrientDB	- Ninguno	- Ninguno	- Sin especificar	- Flexibilidad del Esquema - Lenguaje de consultas - *Sharding* - Respaldos - Modelos múltiples - Arquitectura - Escalabilidad

Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark				
Bases de Datos Evaluadas	**Sets de Datos**	**Algoritmos**	**Ambiente de validación**	**Métricas**
- Neo4j - Jena - HypergraphDB - DEX	- Sets sintéticos generado con algoritmo R-MAT	- Operaciones de inserción - Búsqueda de nodos con peso mayor - K-Neighborhood - Recorrido completo de grafo	- Sistema: Quad Core Intel Xeon E5410 a 2.33 GHz - RAM: 11 GB - Almacenamniento: LFF 2.25 TB	- Desempeño en inserciones, y búsquedas